Automated Knowledge Generation with Persistent Video Surveillance

THESIS

Daniel T. Schmitt, Major, USAF

AFIT/GCS/ENG/09-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/09-06

Automated Knowledge Generation with Persistent Surveillance Video

THESIS

Presented to the Faculty

Department of Electrical Engineering and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Daniel T. Schmitt, B.S.C.S.

Major, USAF

March 2009

AFIT/GCS/ENG/09-06

Automated Knowledge Generation with Persistent Surveillance Video

Daniel T. Schmitt, B.S.C.S.
Major, USAF

Approved:

/signed/	23 Feb 2009
Lt Col Stuart H. Kurkowski, PhD (Chairman)	date
/signed/	23 Feb 2009
Steven K. Rogers, PhD (Member)	date
/signed/	23 Feb 2009
Maj Michael J. Mendenhall, PhD (Member)	date

AFIT/GCS/ENG/09-06

Abstract

The Air Force has increasingly invested in persistent surveillance platforms gathering a large amount of surveillance video. Ordinarily, intelligence analysts watch the video to determine if suspicious activities are occurring. This approach to video analysis can be a very time and manpower intensive process. Instead, this thesis proposes that by using tracks generated from persistent video, we can build a model to detect events for an intelligence analyst. The event that we chose to detect was a suspicious surveillance activity known as a casing event. To test our model we used Global Positioning System (GPS) tracks generated from vehicles driving in an urban area. The results show that over 400 vehicles can be monitored simultaneously in real-time and casing events are detected with high probability (43 of 43 events detected with only 4 false positives). Casing event detections are augmented by determining which buildings are being targeted. In addition, persistent surveillance video is used to construct a social network from vehicle tracks based on the interactions of those tracks. Social networks that are constructed give us further information about the suspicious actors flagged by the casing event detector by telling us who the suspicious actor has interacted with and what buildings they have visited. The end result is a process that automatically generates information from persistent surveillance video providing additional knowledge and understanding to intelligence analysts about terrorist activities.

Acknowledgements

I would like to thank my advisor Lt Col Stuart Kurkowski for his insight and advice. I would like to thank my fellow students and those individuals from Woolpert, General Dynamics, and QBase that helped teach me the skills required to interact with GIS information. I would also like to thank my committee, Maj Mendenhall and Dr. Rogers for their help, influence and guidance. Lastly, I would like to thank my wife for her ability to understand and cope with having a husband that has so many demands on his time.

Daniel T. Schmitt

Table of Contents

Page

Abstract iv

Acknowledgements v

Table of Contents vi

List of Figures x

List of Tables xii

I. Introduction 1
- 1.1 Problem Statement 2
- 1.2 Research Goals 2
- 1.3 Assumptions 2
- 1.4 Contributions 3
- 1.5 Thesis Organization 3

II. Background 5
- 2.1 Artificial Intelligence 5
  - 2.1.1 Formal Logic 6
  - 2.1.2 Machine Learning 10
  - 2.1.3 Knowledge Representation 10
  - 2.1.4 Rule-Based Systems (Expert Systems) 12
  - 2.1.5 Inference engines 13
- 2.2 Semantic Web 13
  - 2.2.1 Uniform Resource Identifier (URI) 14
  - 2.2.2 XML 15
  - 2.2.3 XML Schema 17
  - 2.2.4 RDF 18
  - 2.2.5 RDFS 21
  - 2.2.6 Taxonomies 22
  - 2.2.7 Ontologies 22
  - 2.2.8 Similarities and Differences of Vocabularies, Taxonomies, and Ontologies 26
  - 2.2.9 Evolution of Ontological Languages 27
  - 2.2.10 OWL 28
  - 2.2.11 The Importance of Common Meaning in Building a Semantic Web 30

Page

2.2.12 An Example Semantic Web Project: CS AKTiveSpace . . . 31
2.2.13 Semantic Web Applications . . . 31
2.2.14 Summary of Semantic Web . . . 39
2.3 Computer Video Indexing . . . 40
2.3.1 Abstracting Multimedia Content . . . 40
2.3.2 Matching Techniques . . . 43
2.3.3 Learning Methods Applied to Multimedia . . . 44
2.4 Geographical Information Systems and Databases . . . 44
2.4.1 ArcGIS . . . 44
2.4.2 Oracle Spatial . . . 45
2.4.3 GIS data resources . . . 45
2.5 Surveillance Domain Knowledge . . . 47
2.5.1 Suspicious Activities . . . 48
2.5.2 Use of Social Networks in Surveillance Activities and Counterterrorism . . . 49
2.6 Summary . . . 51

III. Casing Event Detection Methodology . . . 52
3.1 Overview . . . 53
3.2 Turn Detection . . . 53
3.2.1 Problem Definition . . . 54
3.2.2 Workload . . . 54
3.2.3 Results . . . 55
3.3 Point Related Semantics . . . 56
3.3.1 Problem Definition . . . 56
3.3.2 Workload . . . 56
3.3.3 Results . . . 56
3.4 Detecting Building Surveillance Activities . . . 57
3.4.1 Problem Definition . . . 59
3.4.2 System Boundaries . . . 59
3.4.3 System Services . . . 61
3.4.4 Workload . . . 62
3.4.5 System Parameters . . . 62
3.4.6 Factors . . . 65
3.4.7 Evaluation Technique . . . 66
3.4.8 Experimental Design . . . 66
3.5 Casing Event Detection Methodology Summary . . . 66

Page

IV. Casing Detection Results and Analysis 68
4.1 Results and Experiment Discussion 68
4.1.1 Preliminary Results 68
4.1.2 Revised Approach 68
4.1.3 Experiment #1: Determining *Loop Distance* and *Intersection Semantics* 70
4.1.4 Experiment #2: Varying *Time Match Threshold* and Improving Performance Time 71
4.1.5 Experiment #3: Varying *Intersection Distance* and Improving Performance Time 71
4.1.6 Experiment #4: Testing against more data . . . 73
4.1.7 Analysis 74
4.1.8 Casing Event Detection Results Summary . . . 75
4.2 Extending Casing Events Detection 75
4.2.1 Time Duration 76
4.2.2 Target Building 76
4.2.3 Number of Passes 80
4.2.4 Address Lookup 81
4.2.5 Use of Extended Casing Results at Higher Knowledge Levels 81
4.3 Summary of Casing Event Detection 82

V. Automated Social Network Construction Methodology 83
5.1 Problem Definition 83
5.2 Workload 84
5.3 Parameters and Definitions 85
5.3.1 Definition of a stop 85
5.3.2 Defining vehicle-to-building links 85
5.3.3 Defining vehicle-to-vehicle links 86
5.4 Variables 87
5.5 Social Network Construction Methodology Summary . . 88

VI. Social Network Construction Results 90
6.1 Implementation 90
6.2 Results 91
6.3 Enhancements 93
6.4 Analysis 94
6.5 Summary 96

Page

VII. Conclusions . . . 97
7.1 Research Goal . . . 97
7.2 Results . . . 97
7.2.1 Casing Event Detection . . . 97
7.2.2 Social Network Construction . . . 98
7.2.3 Results Summary . . . 100
7.3 Research Contributions . . . 101
7.4 Future Work . . . 102
7.4.1 Source Data . . . 102
7.4.2 Event Detection . . . 102
7.4.3 Extending Network . . . 103
7.5 Summary . . . 104

Appendix A. Turn Detection . . . 105
A.1 Methodology . . . 105
A.1.1 Goal . . . 105
A.1.2 Approach . . . 106
A.1.3 System Boundaries . . . 106
A.1.4 System Services . . . 106
A.1.5 Workload . . . 108
A.1.6 Performance Metrics . . . 111
A.1.7 System Parameters . . . 111
A.1.8 Factors . . . 115
A.1.9 Evaluation Technique . . . 117
A.1.10 Experimental Design . . . 117
A.1.11 Turn Detection Methodology Summary . . . 118
A.2 Turn Detection Results and Analysis . . . 119
A.2.1 Algorithm . . . 119
A.2.2 Results . . . 124
A.2.3 Analysis . . . 126
A.2.4 Turn Detection Summary . . . 127

Appendix B. Oracle Spatial Semantic Tagging Code . . . 128
B.1 Database Organization . . . 128
B.2 Road Semantics . . . 128
B.3 Parking Lot Semantics . . . 130
B.4 Turn Semantics . . . 131
B.5 Intersection Semantics . . . 133

Bibliography . . . 135

List of Figures

Figure Page

2.1. The Smart Data Continuum [69] 14

2.2. The Semantic Web Layer Cake [82] 16

2.3. An Example XML coded document 17

2.4. Evolution of data fidelity [69] 18

2.5. An example taxonomy using UML [39] 23

2.6. The ontology spectrum: Weak to strong semantics [69] 24

2.7. A *Protégé* screen shot [45] 30

2.8. A Home Depot Online Buying Experience [2] 33

2.9. Mashups over time [44] 35

2.10. Breakout of Mashups on Web [36] 35

2.11. RacerPro application interface layers [46] 38

2.12. An example ArcGIS screenshot showing the road network and distances from hospitals in the area [5] 45

2.13. An Oracle SQL Developer screenshot [35] 46

2.14. An example social graph [14] 50

3.1. Traditional flow of processing in a visual surveillance system [87] 53

3.2. Evolution of data fidelity as it relates to Persistent Surveillance Video 54

3.3. A sample overhead surveillance shot [63] 55

3.4. Pictorial results of semantic interpretation of GPS tracks . . . 58

3.5. The finite automata used to detect casing events 60

3.6. A block diagram of the Casing Event Detection System (CEDS) 61

3.7. An example of five right turns used to help define *Complete Loop Distance* 64

4.1. A revised finite automata used to detect casing events 69

Figure Page

4.2. A three turn approach to detecting casing events. Searching for a location match traces forward from the 3rd turn and backward from the 1st turn comparing all points in intersections until one is found less than the *Complete Loop Distance Error Tolerance*. The *Complete Loop Distance* is depicted as the distance between the past and future tracks. 70

4.3. A screen shot of casing detection and determining which building is the target of surveillance 80

5.1. An example of a vehicle-to-vehicle interaction and resulting graph representing a social network 84

6.1. Social network graph of vehicle tracks and buildings 92

6.2. Radial view of social network graph 93

6.3. Social network with buildings organized in geo-space 94

6.4. Area of interest of kidnaping event 95

6.5. Social graph with buildings links hidden 96

7.1. Social network with buildings organized in geo-space 99

7.2. Area of interest of kidnaping event 100

A.1. A sample overhead surveillance shot over Ohio State University [63] 105

A.2. A block diagram of the Turn Detection System 107

A.3. Tracks viewed through the GIS 110

A.4. Example of system parameters in the Turn Detection System . 113

A.5. Block diagram of the sequence of steps in detecting turns . . . 119

B.1. The table holding the basic GPS data 129

List of Tables

Table Page

2.1. Properties of logical systems [76] 6

2.2. Example Propositions 7

2.3. Common Logical Connectives used in Propositional Logic [76] . 7

2.4. The language of First-order Predicate Logic 8

2.5. Definition of terms used in Horn logic [76] 9

2.6. N3 and N-Triple Notation Descriptions 19

2.7. Dublin Core Vocabulary Version 1.1 [16] 20

2.8. Ontological levels [69] 25

3.1. Parameters that affect the performance of the CEDS 63

3.2. Levels of the *Centerline to Road Distance* factor to be tested . 65

3.3. Levels of the *Complete Loop Distance Error Tolerance* factor to be tested 65

3.4. Levels of the *Intersection Semantics Consideration* factor to be tested 66

3.5. Configuration of the CEDS under test 66

4.1. Levels of the *Time Match Threshold* factor to be tested 69

4.2. Results of Casing Detection varying *Loop Distance* and *Intersection Semantics* 71

4.3. Results of Casing Detection varying *Time Matching* 72

4.4. Results of Casing Detection varying *Intersection Distance* . . . 72

4.5. Results of Casing Detection varying *Intersection Distance* . . . 73

4.6. Results of Casing Detection on 17 hours and 8 tracks 74

4.7. Number of casing event detections by vehicle track 74

5.1. Time and Distance Definitions for a stop, vehicle-to-vehicle link, and vehicle-to-building link 87

5.2. Definition of Social Network Variables 88

Table Page

5.3. Behavior of Social Network variables in various environments . 89

7.1. Results of Casing Detection 97

A.1. The events that will be detected by the algorithm 106

A.2. The possible outcomes of running the Turn Detection System . 108

A.3. Parameters that affect the performance of the TDS 114

A.4. Levels of the *Before Event Distance* factor to be tested 115

A.5. Levels of the *After Event Distance* factor to be tested 115

A.6. Levels of the *Turn Degree* factor to be tested 116

A.7. *Matching Threshold* time between a detection and the truth of an event occurring 117

A.8. Threshold of *Speed Differential* between leaving a turn and entering a turn 117

A.9. Configuration of the TDS under test 117

A.10. Results set for one combination of factors to demonstrate equation results 124

A.11. Top results of Turn Detection without using a speed differential 125

A.12. Initial results of Turn Detection using a speed differential . . . 126

A.13. Top results of Turn Detection using a speed differential 127

B.1. Listing of relevant tables in the Semantic tagging of GPS data 129

B.2. SQL code for updating road semantics 130

B.3. SQL code for updating parking lot semantics 131

B.4. SQL code for updating turn semantics 132

B.5. SQL code for updating road intersection semantics 134

Automated Knowledge Generation with Persistent Surveillance Video

I. Introduction

Today's global war on terror is a war of intelligence. The ability to know where terrorists are hiding in plain sight is the key to winning the war on terror. The intelligence that we have increasingly become reliant on in this war are persistent video platforms. They are the Predator unmanned aircraft [22], the Global Hawk [23], and other unmanned aerial vehicles that allow us to view real-time video of the enemy and observe the enemy's activities. The value that these platforms bring to the war has stimulated the need to grow the capability by increasing the number of platforms available, and the number of crews to man them [4] [29]. This increased capability is helping to win the war on terror.

With the increase in persistent video platforms, comes the increase in the amount of video that we have for intelligence analysts to analyze. With hundreds of thousands of flights with persistent video platforms, we have generated tens of thousands of hours of video for analysts to dig into [51]. Often the methods used are simple fast forward and rewind capabilities in order to track enemy activities and movements. Video processing with computers can increase the capabilities beyond fast forward and rewind capabilities to event detection and data extraction. With thousands of hours of video, the benefit of event detection and data extraction tools would increase the amount of knowledge we derive from persistent video, allow us to analyze more video in less time, and further accelerate persistent video as a significant technology assisting in the war on terror.

1.1 Problem Statement

The problem addressed in this thesis is to reduce the information gap that exists between intelligence analysts and the persistent video they analyze allowing us to react faster to new intelligence, and generate more knowledge from existing video intelligence.

1.2 Research Goals

The goal of this research is to generate knowledge from persistent video without the constant intervention of a human. The more information that computers can extract automatically from the video, the less analyst time is required to analyze it, resulting in more video analyzed with fewer resources. To do this we will build from syntactic lower-level data points (vehicle tracks), to understanding the behavior of vehicles (knowing where they stop and turn), detecting casing events, and relating vehicles that had suspicious casing detections to other vehicles. In this way, we intend to provide an intelligence analyst with much more information than simply video to analyze. We intend to provide them with specific vehicles that appear suspicious, time indexes to the video for these vehicles, known associates to the suspicious vehicles, and buildings that were visited by the suspicious vehicle.

1.3 Assumptions

This thesis assumes the following in order to detect casing events and build social networks of interactions:

1. There exists a way to track the movement of vehicles over time in a set area. That tracking can produce a series of waypoints (a track) denoting the position of the vehicle over time.
2. Geographic Information System (GIS) Layers representing roads and buildings exist for the areas of interest. If layers do not already exist then they can be

built using automated tagging techniques, or at the very minimum, manually built.

1.4 Contributions

The contribution of this research is in closing the gap between raw data collection and human knowledge and understanding of persistent video. Where GIS systems can help somewhat visually and geographically orient someone to actions that occur, and allow us to correlate them in geo-space, more analysis is required to bring in time-dependent data to correlate it in both geo-space and time-space. This research will focus on defining ways for geo-space and time-space to interact, and use example scenarios to show the benefit to the area of intelligence and surveillance.

1.5 Thesis Organization

Chapter II begins by providing a background of computer techniques that will be applied to generate knowledge from persistent video data. Section 2.1 discusses the background of Artificial Intelligence and the reasoning engines that will be applied to generate knowledge from data. Section 2.2 discusses background on the Semantic Web, an approach in generating knowledge from large, global data sets. Section 2.3 discusses past work that has been done to syntactically and semantically index video. Section 2.4 discusses Geographical Information Systems (GIS) and the capabilities that exist in relating information geographically. Lastly, in Section 2.5 we discuss the area of surveillance, suspicious activities of interest, and social network tools that are used to help analyze terrorist networks.

Chapter III develops a methodology to define and detect casing events where vehicles drive suspiciously around buildings that are being evaluated for future attacks. The first steps to define a casing event is done in Section 3.2 and fully detailed in Appendix A to detect turns of vehicles being tracked from persistent video. Additional information is also needed from the context of where vehicles are driving and that is explained and incorporated in Section 3.3. Section 3.4 details the methodology that is used to take turn detection and semantic context and build a model for detecting casing events. In Chapter IV we present the results of casing event detection. Upon presenting the results (Section 4.1), more information is added to the casing event to include the suspected building, the duration of the event, and the address of interest which is discussed in Section 4.2.

In Chapter V we discuss a different methodology aimed at detecting and constructing a social network of interactions from GIS data and vehicle tracks. Chapter VI presents the results of social network construction and presents several ways that the information can be visualized to enhance the usefulness for intelligence analysts.

Finally, in Chapter VII we tie the casing event detections with the social network together as helpful methods in analyzing and processing persistent video. Furthermore, we present recommendations for future research in knowledge generation from persistent video.

II. Background

In this chapter, we will discuss the background of Artificial Intelligence, Semantic Web, image processing, Geographical Information Systems, and basic surveillance tactics as it relates to knowledge generation. This information will provide a basis from which we will build a methodology for extracting knowledge from vehicle tracks and GIS layers.

2.1 Artificial Intelligence

Artificial Intelligence is a very broad field as it relates to Computer Science. Its roots trace back to Greek philosopher Aristotle describing humans as the only "rational animal" on the earth. He also established many precedents in the field of logic and began the process of defining syllogisms, a process that was further extended in the 17th century by Leibniz [26]. Leibniz was a German mathematician who in addition to developing the math of calculus initiated an effort to represent human logic and reasoning in a mechanical and symbolic way. Later in the 19th century, George Boole developed Boolean logic which opened the door for the use of mechanical rules to carry out logical deductions [76]. With logic defined in terms of mechanical rules, a computer could be programmed to make logical deductions [64]. There is a controversial difference of opinion over the nature of human intelligence and whether it is possible to construct an artificial source of intelligence. The differences in points of view primarily center on the definition of intelligence. As a result artificial intelligence has focused on certain attributes that most people would agree reflect the concept of intelligence including the ability to learn, the ability to assimilate, the ability to organize and process information, and the ability to apply knowledge to solve complex problems [64]. It was not until the 1940s and 1950s when English mathematician Alan Turing and John McCarthy of MIT began exploring intelligent machines that the phrase "Artificial Intelligence" was coined. This spawned much of the research efforts involving computers that we know of as "Artificial Intelligence" today.

2.1.1 Formal Logic. Much of the field of Artificial Intelligence is based on the rules established in formal logic. Logic is the study of the principles of valid inference and demonstration [76]. As a formal science, logic investigates and classifies the structure of statements and arguments, both through the study of arguments in natural language and through the study of formal systems of inference. Traditionally, logic has been considered a branch of philosophy, as a part of the classical trivium of grammar, logic, and rhetoric. The development of formal logic and its implementation in computing machinery have become fundamental to computer science as they apply to Artificial Intelligence [76].

Formal logic is the study of inference with purely formal content, where that content is made explicit. Formal logic gives us the tools to define logical systems which are in essence representations of the real world under the assumptions in the logical system.

Logical systems can be shown to have certain properties that describe their form. These properties are listed in Table 2.1. We often focus on building sound, consistent logical systems so that we can always deduce new truth from other truths and program computers to do that deduction for us.

Table 2.1: Properties of logical systems [76]

Consistency	None of the theorems of the system contradict one another.
Soundness	A system's rules of proof will never allow a false inference from a true premise. If a system is sound and its axioms are true then its theorems are also guaranteed to be true.
Completeness	There are no true sentences in the system that cannot, at least in principle, be proved in the system.

Since the nineteenth century, formal logic has been studied in the context of mathematics, where it was usually referred to as symbolic logic. Symbolic logic is the study of symbolic abstractions that capture the formal features of logical inference. Symbolic logic is often divided into two branches, propositional logic and predicate logic [76].

2.1.1.1 Propositional Logic. A propositional logic system is a formal system in which formulae representing propositions can be formed by combining atomic propositions using logical connectives, and a system of formal proof rules allows certain formulae to be established as "theorems" [76]. Our goal in defining this formal system is to develop a logic system that is not as ambiguous as a natural language such as the English language is. Atomic propositions, also called "propositions," are declarative statements which one can argue is true or false [76]. We often use literals such as p or q or any other lower case letter to represent such statements. They can be defined as needed to demonstrate proof. For example, we could define p to represent the propositional statement "It is raining outside." This is a propositional statement because it can be either true or false. Table 2.2 shows some example propositional statements.

Table 2.2: Example Propositions

Symbol	Meaning
p	It is raining outside.
q	I have $20 in my wallet.

We string propositions together with logical connectives to build compound statements. Some of the more common logical connectives and their description are listed in Table 2.3 along with some examples and interpretations based on the examples in Table 2.2.

Table 2.3: Common Logical Connectives used in Propositional Logic [76]

Name	Symbol	Meaning	Example	Interpretation
Negation	$\neg$	NOT	$\neg p$	It is not raining outside.
Disjunction	$\vee$	OR	$p \vee q$	Either it is raining outside, or I have $20 in my wallet.
Conjunction	$\wedge$	AND	$p \wedge q$	It is raining outside and I have $20 in my wallet.
Implication	$\rightarrow$	IF-THEN	$p \rightarrow q$	If it is raining outside, then I have $20 in my wallet.

These symbols allow us to represent more complex statements while maintaining the preciseness of the logical language. Each symbol comes with a method to evaluate truth or falsity associated with its use which are called proof rules. For example, the statement "It is raining outside and I have \$20 in my wallet" is only true of both p (It is raining outside) and q (I have \$20 in my wallet) are true and false if either p or q are false. Propositional logic then takes natural language statements, translates them into symbolic logic statements, then asks whether new true statements can be formed from those statements. These are known as arguments, and the method for proving them true is known as a proof. To accomplish a proof one uses the proof rules of the particular logical connectives to determine if a new proposition is true or not. With this system, we begin to form a method for evaluating new truths from old truths in a precise systematic fashion that can be accomplished by a computer.

2.1.1.2 First Order Logic. First-order logic (FOL) extends propositional logic with the concepts of predicates and quantification. FOL goes by many names, including: first-order predicate calculus (FOPC), the language of first-order logic, first-order predicate logic (FOPL), or predicate logic. Unlike natural languages such as English, FOL uses a wholly unambiguous formal language interpreted by mathematical structures [76].

Predicates are statements that can be applied and tested to many literals. Table 2.4 gives some examples of a predicate and the two types of quantification: there exists(∃), and for all (∀).

Table 2.4: The language of First-order Predicate Logic

Concept	**Usage Example**	**Meaning**
Predicate	$Man(x)$	x is a Man
∃	$\exists x Man(x)$	Of all the items in x, at least one is a Man
∀	$\forall x Man(x)$	All items in x are men.

Take for example the following sentences: "Socrates is a man", "Plato is a man". In propositional logic these will be two unrelated propositions, denoted for example by p and q. In first-order logic however, both sentences would be connected by the same property: $Man(x)$, where $Man(x)$ means that x is a man. When x=Socrates we get the first proposition, p, and when x=Plato we get the second proposition, q. Such a construction allows for a much more powerful logic when quantifiers are introduced, such as "for every x, if $Man(x)$, then $Mortal(x)$" [76]. This could be written as $\forall x(Man(x) \rightarrow Mortal(x))$.

The addition of predicates and quantifiers to FOL makes it a richer language allowing us to express more about the natural world than propositional logic does.

2.1.1.3 Horn Logic. Building on propositional logic and first order predicate logic, Horn logic is a organization of logic into Horn clauses in order to simplify the computational complexity of deducing new truths [76]. A Horn clause is a clause (a disjunction of literals) with at most one positive literal. Horn clauses are named for the logician Alfred Horn, who first pointed out the significance of such clauses in 1951 [75]. Table 2.5 defines several terms that are used in Horn logic.

Table 2.5: Definition of terms used in Horn logic [76]

Definite Clause	A Horn clause with exactly one positive literal.
Goal Clause	A Horn clause with no positive literals.
Horn formula	A conjunctive normal form formula whose clauses are all Horn. A conjunction of Horn clauses.

The following is an example [76] of a (definite) Horn clause:

$\neg p \vee \neg q \vee \cdots \vee \neg t \vee u$

Such a formula can also be written equivalently in the form of an implication:

$(p \wedge q \wedge \cdots \wedge t) \rightarrow u$

The resolution of a Horn clause is quick because of the form it is expressed in. Relating this back to the example, this means if in evaluating $(p \wedge q \wedge \cdots \wedge t)$ and we find a $p, q, \cdots, t$ that is false, we can immediately conclude that u will be false, without having to test all terms $p, q, \cdots, t$. What makes Horn clauses important to automated reasoning is that resolution of two Horn clauses results in a Horn clause. In addition, the resolution of a goal clause and a definite clause is also a goal clause. This resolution allows the chaining of goal clauses and Horn clauses together to deduce additional truths not previously generated. The process of reduction on statements as Horn clauses and goal clauses, if done before specific cases are tested, can lead to greater efficiencies in proving a theorem (represented as a goal clause) [76]. The resolution of a goal clause with a definite clause to produce a new goal clause is the basis of the logic used to implement logic programming and the programming language Prolog [1] along with forming the logic foundation for all other deductive reasoning engines that are discussed in Section 2.1.5.

2.1.2 Machine Learning.

Machine learning is a broad subfield of Artificial Intelligence (AI) concerned with the design and development of algorithms and techniques that allow computers to "learn". At a general level, there are two types of learning: inductive, and deductive [65]. Inductive machine learning methods extract rules and patterns out of massive data sets, while deductive are programmed with rules ahead of time and asked to apply them to data as it is fed. Deductive applies the rules in accordance with the structure as it is defined in formal logic. Both deductive and inductive learning are used in computer science. Inductive is used to determine what rules might apply over a given set of data. Deductive then applies those rules, or other implicitly defined rules to determine whether new statements (inferences) are true.

The major focus of machine learning research is to extract information from data automatically, by computational and statistical methods. Hence, machine learning is closely related not only to data mining and statistics, but also theoretical computer science [65].

2.1.3 Knowledge Representation.

Knowledge representation is a section of AI that focuses on the design and use of languages and systems that represent knowledge about the world. Knowledge representation is central to AI research because many of the problems machines are expected to solve require extensive knowledge about the world [41]. Knowledge representation is a means for expressing and using semantic information or knowledge about the world with the additional requirement that the use of that knowledge be used for intelligent reasoning and compute efficiently when modeled and run on a computer [69].

Knowledge representation's underpinnings were developed in the 1960s and 1970s beginning with the idea of a semantic network by Quillian in 1968 [78]. A semantic network is a network structure (often a directed acyclic graph) used for the expression of semantics using a node-and-link representation of concepts and their relationships. We call these "knowledge maps" today. The general shortfall of semantic networks is that they were ad hoc, heterogeneously structured and represented with their semantics dependent only on the particular researchers who developed them and the particular systems they were used on. They were not generic in any way and thus not portable to other applications. Also, their reasoning methods were based on the particular implementation strategy, not on a formal language [69]. Thus semantic networks, while useful, had limitations on their use and growth. It was not until the late 1970s and early 1980s that the knowledge representation community began to formalize a consistent notion of a semantic network and develop a logic-based language for representing knowledge with the introduction of KL-ONE as a frame-based language [67]. A frame-based language is a language for representing knowledge based on frames, which are data structures for representing concepts and relations. Frame-based systems are close to the object-oriented modeling paradigm, as knowledge of a domain is centered on entities (objects) of the domain. The frame perspective is different from other languages in the fact that they are based on logical axioms making frame-based representations equivalent to logic-based representations. Current day knowledge representations like Web Ontology Language (OWL) discussed in Section 2.2.10 are built in this manner [69]. Frame-based systems led to the development of description logics. A description logic is a language for representing knowledge that has a syntax and semantics; a syntax for building descriptions and a semantics that defines the meaning of each description. Description logics formalized the nature of knowledge representation logically and provided sound, tractable reasoning methods to reason about the encoded semantics [69]. This allows a reasoning to occur over the semantic representations as its founded on a system of formal logic from which we can deduce truth or fallacy from. This sets the stage to allow computers to reason new knowledge based on formal rule sets and other information and data that is related together.

2.1.4 Rule-Based Systems (Expert Systems). An "expert system" can be considered a particular type of knowledge-based system where the knowledge is represented "as it is" [64]. Developed beginning in the 1980s, expert systems are computer applications that make decisions or inferences in real-life situations, which would otherwise be performed by a human expert [41]. They are programs designed to mimic human thinking in deducing causes for constrained problem-solving tasks. They are often built by layering a series of IF-THEN rules combined with reasoning engines that allow a problem to be investigated, decomposed, and solved [64]. An important element of any expert system is its body of knowledge related to its knowledge domain. This information is often extracted from human experts using rules of thumb or other heuristic information. The knowledge provides the basis of what the system "knows" about a knowledge domain and is used to supplement the knowledge of the individual using the system. Just as the common definition of a computer program can be defined as [64]:

$$algorithm + data\_structures = program$$

A similar analogy can be applied to expert systems where [64]:

$$inference\_engine + knowledge = expert\_system$$

2.1.5 Inference engines. Inference engines, also called reasoning engines, are the programs or agents that deduce new knowledge from already specified, pre-programmed information. There are two main approaches taken in the execution of inference engines: specialized algorithms (Problem Solving Methods), and logic-based inference engines [64]. Specialized algorithms, or Problem Solving Methods (PSMs) as they are also called, are small algorithms that perform inferences within expert systems. A PSM specifies which inference and actions have to be carried out to achieve the goal of a task and define the control and data flow between subtasks. PSMs are often very specific, and each new task requires a new PSM to specify it [64].

Logic-based inference engines are considerably more powerful. The engine controls overall execution of a set of rules by searching through a knowledge base and attempting to pattern-match facts or knowledge to the antecedents of rules. If a rule's antecedent is satisfied, the consequence of a rule can be executed. The priority of which rules are evaluated first and which consequences are executed in what order is set by the expert system in such a way to improve the speed of reasoning [64]. It is in this manner that a basis of information can be run against an inference engine in order to deduce new information based on the rule system in the engine. The fact that inference engines are based on sound formal logic makes the inferences found, new truths, thus extending the knowledge of the system.

Inference engines can be forward or backward chaining. In forward chaining, the inference engine is given data with which it chains forward to reach a conclusion. In backward chaining, an inference engine is given a hypothesis with which it backtracks to check if a hypothesis is valid or not [64]. Different inference engines that are used commonly today in Semantic Web processing are discussed in Section 2.2.13.2.

2.2 Semantic Web

The Semantic Web is described as "an extension of the current Web [Internet] in which information is given well-defined meaning, better enabling computer and people to work in cooperation" [66]. This "well-defined meaning" is accomplished by adding metadata, taxonomies, and ontologies to allow computers to relate data together to bring about information on its own without constant human intervention. XML (eXtensible Markup Language) and the Resource Description Framework (RDF) are key innovative technologies that allow us to associate the data with metadata, and then begin to form relationships (RDF, taxonomies, and ontologies) with those metadata tags to then allow computers to build information. With increased aggregation and reasoning this information can become knowledge as is demonstrated in the smart data continuum in Figure 2.1.

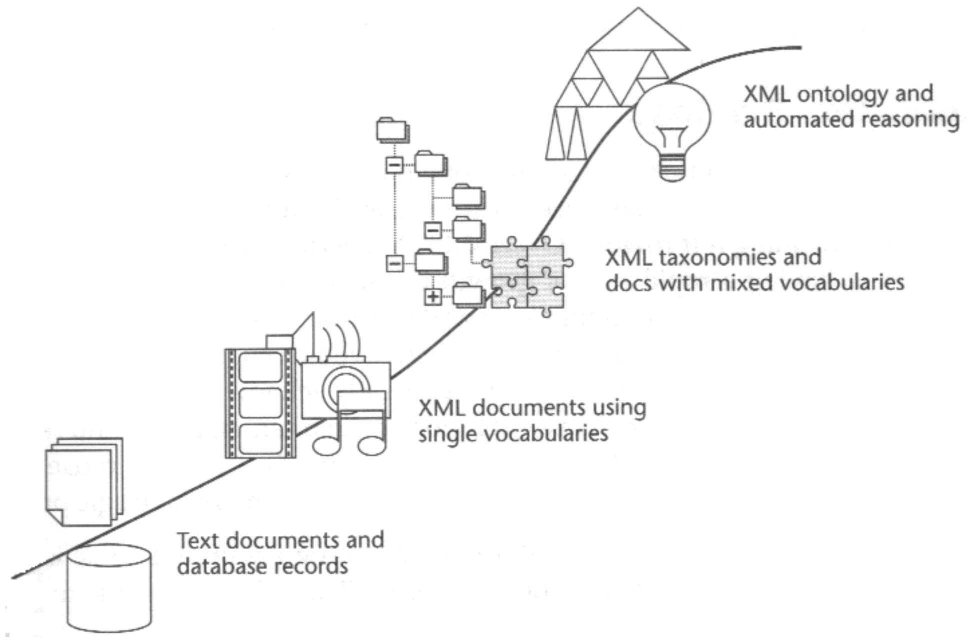

XML ontology and automated reasoning
XML taxonomies and docs with mixed vocabularies
XML documents using single vocabularies
Text documents and database records

Figure 2.1: The Smart Data Continuum [69]

The next few sections provide a description of the technologies that contribute to building a Semantic Web. These include the Uniform Resource Identifier(URI), eXtensible Markup Language (XML), XML Schema, Resource Description Framework (RDF), RDF Schema (RDFS), taxonomies, ontologies, OWL, and Semantic Web applications.

2.2.1 Uniform Resource Identifier (URI). A Uniform Resource Identifier (URI) is a standard syntax for strings that identify a resource. URI is a generic term for addresses and names of objects on the world wide web. There are two kinds of URIs: Uniform Resource Locators (URLs) and Uniform Resource Names (URNs) [69]. URLs, are web addresses on the Internet. A URN is an Internet resource with a name that, unlike a URL, has persistent significance. That means that the owner of the URN can expect that someone else (or a program) will always be able to find the resource. An example URN might be “urn:def:/blue laser” where “def:/” might indicate an available directory of all dictionaries, glossaries, and encyclopedias on the Internet and “blue laser” was the name of a term. A comparable URL would need to specify one specific location for a definition like “http:/www.whatis.com/bluelaser.htm” [57].

2.2.2 XML. XML is the syntactic foundation layer of the Semantic Web [69]. It is a bottom layer that builds upon all items in the web (URIs) as shown in Figure 2.2. It takes data and “tags it” (labels it) so that the data can be given context to the degree that we desire. This tagging can be done to different forms of data to include documents, databases, webpages (HTML), or any other form of information. An example of XML code is provided in Figure 2.3. By tagging the individual strings like “Shimmering Room” as a name associated with a place we begin to associate items of data at a meta data level. This tagging has allowed XML to be the choice set of rules for translating data in the form of databases or documents from one application to another.

XML is written in such a way that it creates a hierarchy of data. While this can be seen more apparent in the indentation provided in Figure 2.3, it is formally defined by the tags that exist within tags. An example of this is the “place” named Shimmering Room that has sub tags of description and travel. Within the scope of the rules of XML we can determine if a document is well formed or not. Examples of a non-well formed XML file is one that has start tags, that do not have end tags (i.e. a <place> with no </place> to terminate it). We cannot however without additional information determine if an XML document conforms to a particular XML structure without additional information known as an XML Schema [69].

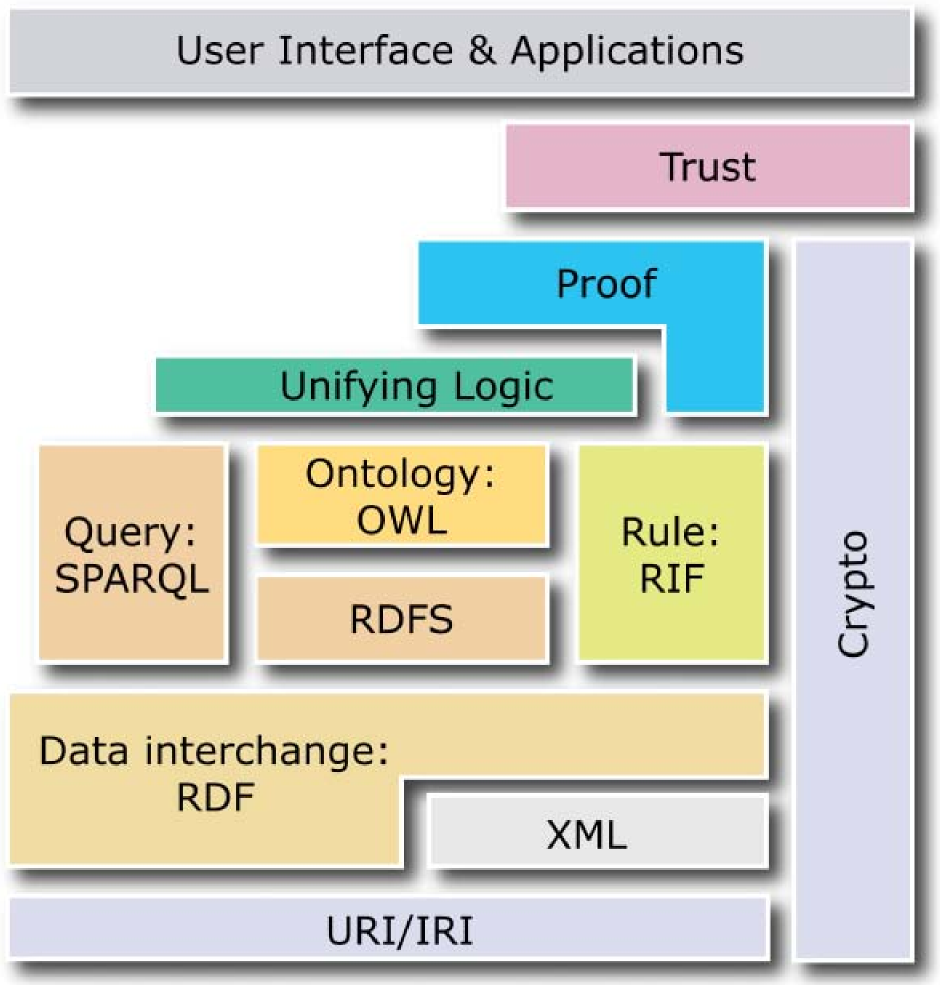
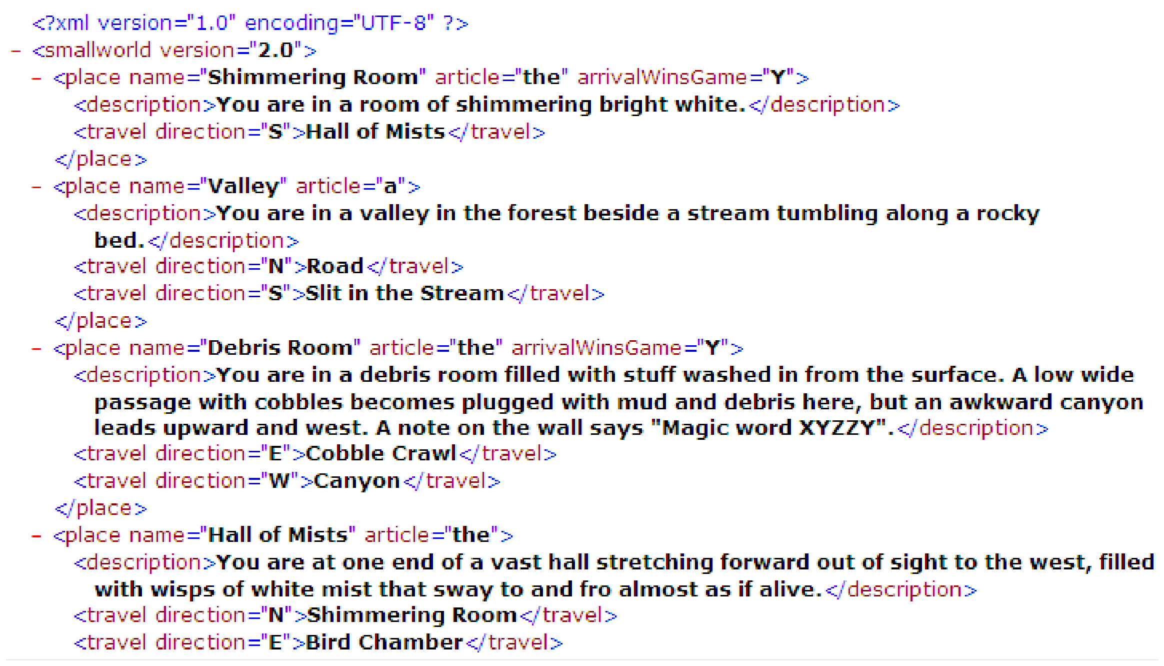


Figure 2.2: The Semantic Web Layer Cake [82]



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <smallworld version="2.0">
  - <place name="Shimmering Room" article="the" arrivalWinsGame="Y">
      <description>You are in a room of shimmering bright white.</description>
      <travel direction="S">Hall of Mists</travel>
    </place>
  - <place name="Valley" article="a">
      <description>You are in a valley in the forest beside a stream tumbling along a rocky
        bed.</description>
      <travel direction="N">Road</travel>
      <travel direction="S">Slit in the Stream</travel>
    </place>
  - <place name="Debris Room" article="the" arrivalWinsGame="Y">
      <description>You are in a debris room filled with stuff washed in from the surface. A low wide
        passage with cobbles becomes plugged with mud and debris here, but an awkward canyon
        leads upward and west. A note on the wall says "Magic word XYZZY".</description>
      <travel direction="E">Cobble Crawl</travel>
      <travel direction="W">Canyon</travel>
    </place>
  - <place name="Hall of Mists" article="the">
      <description>You are at one end of a vast hall stretching forward out of sight to the west, filled
        with wisps of white mist that sway to and fro almost as if alive.</description>
      <travel direction="N">Shimmering Room</travel>
      <travel direction="E">Bird Chamber</travel>
```

Figure 2.3: An Example XML coded document

2.2.3 XML Schema. XML Schema is a definition language that allows an individual to constrain XML documents to a specific vocabulary and hierarchical structure. Continuing the example with the XML code above, it is like having a template document that states every place must have a name. With this schema in hand, we can check various XML documents against the schema to see if they are valid documents for the schema (checking to make sure that every place in the document indeed has a name). XML's overall contributions to the Semantic Web are [69]:

1. XML creates application-independent documents and data.
2. XML provides a standardized syntax for metadata.
3. XML provides a standardized structure for documents and data.
4. XML is a time-proven standard that has been used and tested by industry and proven to be useful in domains other than the Semantic Web.

While these contributions that XML gives us are significant it does not get us where we need to be to build a Semantic Web. Through XML's metadata we have not been able to define things as we tag them. Referring to Figure 2.4, we have created the basis of things that are referred to in Level 1.

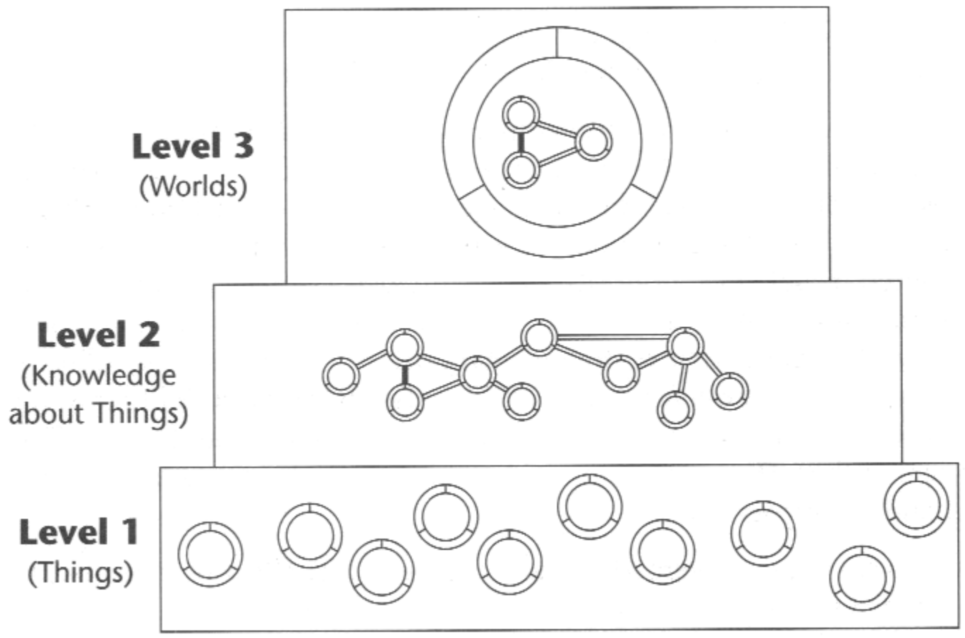


Figure 2.4: Evolution of data fidelity [69]

In order to progress beyond data modeling to knowledge modeling (Level 2) we must begin to relate "things" together. This is where Resource Description Framework (RDF) (Section 2.2.4) and taxonomies (Section 2.2.6) build on XML to begin relating things together into higher level knowledge. Continuing the progression of knowledge, Level 3 is the level of ontologies which we will discuss in Section 2.2.7.

2.2.4 RDF. RDF is a language that builds upon the syntax of XML to build knowledge. RDF is based on the principle that it takes three and only three pieces of information to fully define a single bit of knowledge [82]. These three parts are called the subject, the object, and the predicate. In essence, just as in grammatical context, the subject and the object are both "things" (i.e. URIs). The subject is the doer of the action, and the object is the noun that is being acted upon. The predicate is the action or relationship between the subject and the object. This relationship from subject to object using predicates is known as an RDF Triple [69].

By having these relationships among "things" it is possible to generate new knowledge by exploring the relationships more fully. For example, having the following information:

- *Bob owns Bob's Truck Shop.*
- *Bob is the father of Jenny.*

means that we can generate

- *Jenny's father owns Bob's Truck Shop.*

While this is only a simple example, this example could be extended horizontally and vertically based on the RDF triples that are known to form larger compound statements. Using XML is the official technique to encode RDF information [82]. There are two other common methods for RDF encodings that are used because of their simplicity, they are N3 Notation [82] and N-Triples Notation [82] which are described in Table 2.6.

Table 2.6: N3 and N-Triple Notation Descriptions

N3 Notation	The basic structure of an N3 triple is: *subject object predicate*. The syntax defines that a space should exist between each, and the triple is terminated with a period [82]. An example of the statement "Bob is the father of Jenny" in N3 format would be: *Bob Father_of Jenny*.
N-Triples Notation	N-Triples extends N3 notation. N-Triples allows tabs to be interchanged with spaces in the triple ordering. N-Triples also allows a line to be a comment using a # sign to begin the line [82]. An example of a comment would be: # *This is a comment*. Other than that, N-Triples is exactly the same as N3.

2.2.4.1 Defining a common vocabulary: Dublin Core. It can be seen that a common vocabulary is necessary in order to have RDF triples effectively build upon one another. An organization that has make progress in developing consensus for vocabularies and definitions on the web is the Dublin Core.

The Dublin Core Metadata Initiative's mission is to "provide simple standards to facilitate the finding, sharing and management of information" [16]. On the DCMI website it summarizes the activities of the DCMI as:

> The development and maintenance of a core set of metadata terms (the DCMI Metadata Terms) continues to be one of the main activities of DCMI. In addition, DCMI is developing guidelines and procedures to help implementers define and describe their usage of Dublin Core metadata in the form of Application Profiles. This work is done in a work structure that provide discussion and cooperation platforms for specific communities (e.g. education, government information, corporate knowledge management) or specific interests (e.g. technical architecture, accessibility) [16].

Dublin Core's vocabulary is simple, and that is likely the one reason it is so widely used (located at http://purl.org/dc/elements/1.1/). Its simplicity makes it often the basis for other vocabularies to be added or merged with it, since it contains such foundational definitions. These definitions are listed in Table 2.7.

Table 2.7: Dublin Core Vocabulary Version 1.1 [16]

Name	The label assigned to the data element.
Identifier	The unique identifier assigned to the data element.
Version	The version of the data element.
Registration Authority	The entity authorized to register the data element.
Language	The language in which the data element is specified.
Definition	A statement that clearly represents the concept and essential nature of the data element.
Obligation	Indicates if the data element is required to always or sometimes be present (contain a value).
Datatype	Indicates the type of data that can be represented in the value of the data element.
Maximum Occurrence	Indicates any limit to the repeatability of the data element.
Comment	A remark concerning the application of the data element.

2.2.5 RDFS. After RDF was defined by the World Wide Web Consortium (W3C), it became logical to extend RDF to higher levels of abstraction. If we use an RDF triple to denote a class, class property, and value, then we can create class hierarchies for the classification and description of objects. That is what RDF Schema (RDFS) does for us [69].

RDFS is a simple set of standard RDF resources and properties to help people to create their own RDF vocabularies. In essence it applies the idea of object oriented programming to RDF in order to standardize RDF definitions and encourage their use and reuse. In object oriented programming we use a class to represent a template or blueprint for an object. In defining the class we give it certain attributes (what data elements it is made up of) and methods (what actions can be done with the class). We instantiate classes to become actual working objects in the image class's blueprint. Furthermore, we can create subclasses that inherit the attributes and methods from their parent class (super class). These subclasses can be extended, or changed depending on the definition of the subclass. The Unified Modeling Language (UML) described briefly in Section 2.2.6 is a method of graphically depicting the parent-to-child relationship that exists in classes.

RDFS can also be thought of as similar to relational databases. Relational databases use metadata tags to define the structure of the database. RDFS's metadata does the same thing for RDF and XML information. It provides the resources necessary to describe the objects and properties of a domain-specific schema. This becomes a vocabulary used to describe objects and their attributes within an area of interest [82].

RDFS is popular language to write semantic code in. A 2007 survey of ontology languages used by the community reported 64.9% of individuals surveyed were currently using RDFS. It came in a close second to OWL which was used by 75.9% of individuals surveyed [68]. OWL is discussed in more detail in Section 2.2.10.

2.2.6 Taxonomies.

Taxonomies are ways of classifying or categorizing a set of things in a hierarchy. A definition from the dictionary [52] is :

1. the study of the general principles of scientific classification: SYSTEMATICS
2. CLASSIFICATION; especially : orderly classification of plants and animals according to their presumed natural relationships

Applying this to the Web, we can then say that taxonomies on the Web are classifying information entities in the form of a hierarchy, according to the presumed relationships of the real-world entities that they represent [69]. Thus a taxonomy is a semantic hierarchy in which information entities are related by either the "subclassification of" relationship or the "subclass of" relationship. We use these classifications to further add knowledge and meaning to "things."

These taxonomies on the Web are sometimes diagramed in Universal Modeling Language (UML). An example is provided in Figure 2.5. UML uses the open triangle line connection to denote subclasses and superclasses. Thus, relating to the example in Figure 2.5, a floppy is a magnetic storage which is a component and so on.

While taxonomies are good at classifying "things" and moving up the knowledge hierarchy, they are seen as "weak semantic" systems. Taxonomies are weak semantic systems because the relationship from parent to child (superclass to subclass) is underspecified or ill defined. In a taxonomy, it is possible for a subclass to be defined as a subclass in one instance, and a part of the superclass in another instance. This ambiguity is what makes taxonomies weaker in semantic reasoning and makes it possible for us to deduce knowledge that was not intended. Referring to Figure 2.6 we see that taxonomies are one of the weakest forms for ontologies. We will describe ontologies in more detail in Section 2.2.7.

2.2.7 Ontologies.

The definition of an ontology is often misunderstood because two different disciplines use the word ontology to mean two different things. Ontologies originated in philosophy where it is defined as "a part of metaphysics

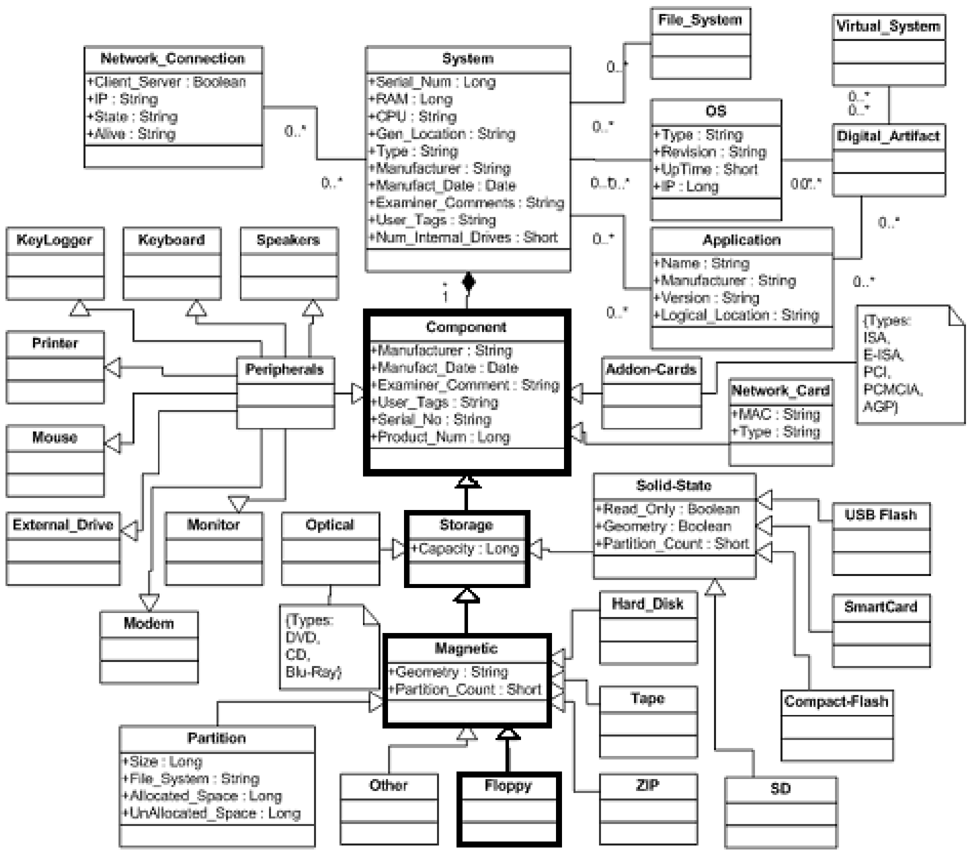


Figure 2.5: An example taxonomy using UML [39]

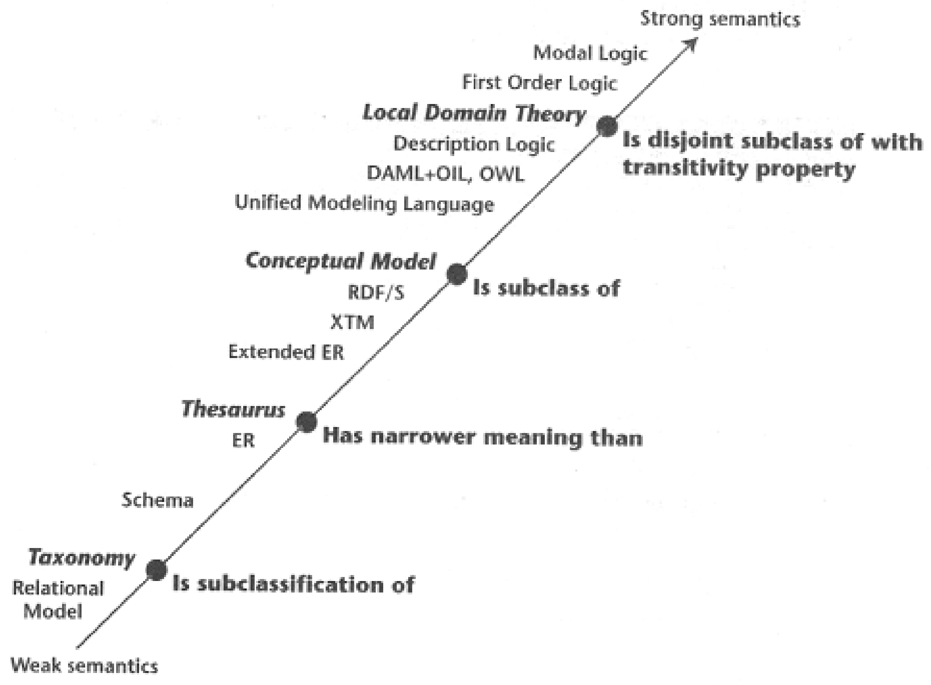


Figure 2.6: The ontology spectrum: Weak to strong semantics [69]

concerned with the nature and relations of being" with a specific theory of the nature of being also being called an ontology [69]. That definition of an ontology still holds since the birth of philosophy during the Renaissance, however, a new definition has emerged with the rise of information technology. A simple definition of an ontology is that an ontology "defines the common words and concepts (the meaning) used to describe and represent an area of knowledge" [69]. To make an ontology we need two things: 1) a vocabulary, and 2) a set of explicit assumptions regarding the intended meaning of that vocabulary.

Vocabularies and the meaning of words within those vocabularies is a key area of defining ontologies. It is essential in a semantic web paradigm to have exactly the same definitions of words. Since this is rarely the case in the real world, committees often meet to discuss and come to agreement on a set of terms and what they mean. This has often been the case when XML, RDF, or other languages are used as a interface between two systems. Each item that is being passed from one application

Table 2.8: Ontological levels [69]

Name of Level	Meta Level to:	Object level to:	Example
Knowledge representation(KR) level	OC Level	N/A	Class, Relation, Instance, Function, Axiom, Rule
ontology concept(OC) level	OI Level	KR Level	Person, Location, Event, Parent, Hammer, etc.
ontology instance(OI) level	N/A	OC Level	Harry S. Truman, Person2554, 1995-Ford Taurus-VIN-67YI84ZZU90

to another has to map unambiguously. Thus defining vocabularies are similar to defining interfaces just in a larger sense and with more stakeholders.

What ontologies give us is an unambiguous language with which we can turn our unambiguous real world definitions into a set of rules or axioms with which a computer can reason the meaning of things. It is important to note that this is our end goal. High end ontology languages are thus backed with tight formal logic making ontologies *machine-interpretable*.

> By *machine interpretable* we mean that the semantics of the mode is semantically interpretable by the machine; in other words, the computer and its software can interpret the semantics of the model directly – *without direct human involvement*. Software supported by ontologies moves up to the human knowledge/conceptual level; humans do not have to move down to the machine level. Interaction with computers takes place at our level, not theirs. This is an extremely important point, and it underscores the value of ontologies [69].

It is important to make distinctions among different representation levels when discussing ontologies. This is necessary because ontologies can be viewed as languages, or vocabularies with accompanying semantics. The levels that ontologies are often divided into are listed in Table 2.8.

The knowledge representation (KR) language level defines the constructs that will be used at the ontology concept (OC) level. KR languages include languages that came before the Semantic Web [69] like Ontolingua, LOOM, Knowledge Interchange

Format (KIF), and UML, and Semantic Web languages like RDF/S, $DAML + OIL$, and OWL. We will discuss ontological languages in detail in Section 2.2.9.

At the OC level, ontologies are defined using the constructs of the KR level. At this level, we are interested in modeling the generic or universal content or domain knowledge. At the ontology instance (OI) level, the constructs are instances of OC level constructs. As a result, this level is concerned with the knowledge base that are assertions about instances or individuals [69].

One might ask how an ontology is constructed. There are two primary methods for constructing ontologies, the push method and the pull method. Pushing is accomplished by web page developers who code their content along with metadata ahead of time in their web documents. Pulling is when web pages are not populated with metadata upon construction requiring webcrawler-like applications to read documents and define relationships from untagged content. The preferred method is pushing because then we can be certain that the meaning is as intended, and not just a result of an applications interpretation of content which could be incorrect.

2.2.8 Similarities and Differences of Vocabularies, Taxonomies, and Ontologies. It may seem that taxonomies, vocabularies, and ontologies are all very similar. That is because they have the following in common [38]:

- They are approaches to help structure, classify, model, and or represent the concepts and relationships pertaining to some subject matter of interest to some community.
- They are intended to enable a community to come to agreement and to commit to use the same terms in the same way.
- There is a set of terms that some community agrees to use to refer to these concepts and relationships.
- The meaning of the terms is specified in some way and to some degree.

They do have differences, however. A vocabulary might not have meaning specified, as it could just be a set of understood terms by the community. On the other hand, a vocabulary could have very detailed definitions for each term. Taxonomies have additional meaning because of the hierarchical links. Thus terms have both definitions of what they are, and definitions from where they are in the hierarchy to include inherited definitions. An ontology is the most commonly used term and is often used naively to refer to taxonomies and vocabularies because ontologies extend the definitions of taxonomies and vocabularies. They tend to refer to things that have a rich and formal logic-based language for specifying the meaning of terms. This formal logic is what gives ontologies additional deductive power and what makes them the most commonly used tool to define semantics for automated reasoning tools. By defining formally the rules in a language that is sound, a computer can apply those unambiguous rules to a data set to move up the data continuum [38].

2.2.9 Evolution of Ontological Languages. Ontological Languages, or Knowledge representation languages as we have referred to them before, have evolved over the last decade. The first significant effort was Simplet HTML Ontology Extension (SHOE) language developed by the University of Maryland which began in 1995 [82]. Work on OIL (Ontology Interface Language) led by the University of Amsterdam and funded by the European Union began in 1997 was released in 2000 [77] [82]. DAML (DARPA Agent Markup Language) was then developed by DARPA (Defense Advanced Research Projects Agency) in 2000 [77]. In December 2000, the two languages (DAML and OIL) were combined into what is known as $DAML + OIL$ [13]. $DAML + OIL$ was later superseded by OWL (Web Ontology Language) which is the current World Wide Web Consortium (W3C) standard ontology language [13]. OWL is in fact a family of languages, OWL-DL, OWL-Lite, and OWL-Full. OWL-DL's and OWL-Lite's semantics are based on Description Logics, which have attractive and well-understood computational properties. OWL-Full uses a different semantic model intended to provide compatibility with RDF Schema (RDFS) [54].

2.2.10 OWL. Because OWL was derived from $DAML + OIL$ and works with XML, RDF, and RDFS, it has become a defacto standard ontology language for the Semantic Web. In fact, a survey done in 2007 listed OWL as the most widely used ontology language with 75.9% of individuals surveyed saying they have used it [68]. When OWL was being developed it had the following design goals [82]:

- Ontologies must be sharable, so that more than one business within a particular business domain could use the same ontology defined for that domain.
- Evolving ontologies should be given version numbers and the schema defining the ontology given a separate URI for each version.
- Ontologies must be interoperable.
- Inconsistencies in ontologies must be detected automatically to prevent them from occurring.
- Ontologies must balance expressivity and scalability.
- Ontologies must be consistent with other standards.

These goals provided the basis for what has become a powerful language to express semantics in the web.

OWL adds capabilities to that of RDF and RDFS. One such capability is the ability to create local range restrictions. In RDF and RDFS, we could only give one range for a property. In many cases the range for a property should vary based on what class is in the domain. As an example, we might have a property to a class "Person" called "eats." We would want to constrain "eats" with a range restriction that it would have to come from something of class "Food." It follows that if we then created a subclass of a "Person" called "Vegetarian" that we would further want to constrain the "eats" relationship. There does not exist a way to do this in RDF or

RDFS. In OWL, we can leave “eats” unchanged, but in the “Vegetarian” class restrict it to only eating “Vegetarian Food,” a subclass of “Food” [77].

OWL also introduces basic set functionality like union, intersection, complement, and disjointedness. As a result we can define classes and subclasses as mutually exclusive (Ex. a “Person” can be either “Alive” or “Dead”). Such disjointedness was not supported or defined in RDF or RDFS [77].

OWL also introduces the concept of cardinality. It allows the ontology author to define maximum and minimum cardinalities on instances. Thus in OWL we can require “twins” to be exactly two children, and “bilingual” to have at least two languages [77].

Because of similarities in the way classes are defined in RDFS and OWL, one might ask when it is more appropriate to code in one or the other. Shelly Powers’, author of Practical RDF, recommendation is:

> If you are defining a fairly simple vocabulary primarily for your own use, and if you are concerned primarily with the striped nature of RDF/XML, you will most likely want to just define your vocabulary in RDF and RDFS. However, if you are documenting a model of a specific domain and you hope to encourage others to use it and, best of all, be able to use the data to make sophisticated queries, you are going to want to use OWL to take advantage of its many inferential enhancements [82].

2.2.10.1 Ontology Editors. With the growth of ontologies and OWL has come a growth in tools that help manipulate and code ontologies. The most widely used ontology editor is one developed by Stanford University called *Protégé* [45]. In a 2007 survey it was listed as the #1 ontology editor with 68.2% of individuals surveyed responding that they were using it currently with the #2 editor SWOOP being used by only 13.6% of individuals surveyed [68]. The *Protégé* platform supports two applications for modeling ontologies: the $Protégé-Frames$ and $Protégé-OWL$ editors. *Protégé* ontologies can be exported into a variety of formats including RDFS, OWL, and XML Schema. Written in Java, *Protégé* is extensible, and provides a plug-andplay environment that makes it a flexible base for rapid application development [45].

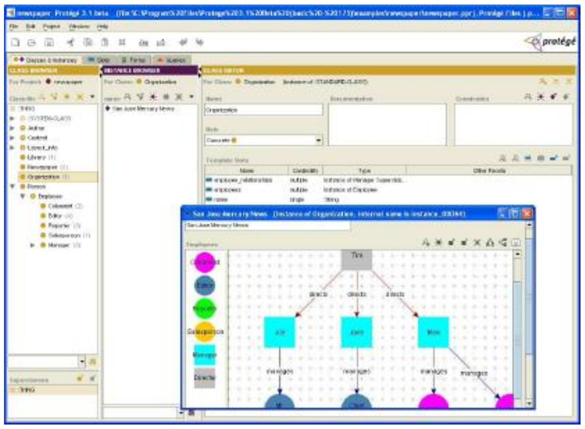


Figure 2.7: A *Protégé* screen shot [45]

2.2.11 The Importance of Common Meaning in Building a Semantic Web. There have been some individuals that believe that it takes more than new technologies and coding standards to bring about the Semantic Web; it takes agreement on the meaning of words. David Moschella stated such an argument in "Semantic Applications, Or Revenge of the Librarians" [80]. He posits that with the addition of metadata and the progress of individual industries toward forming a common language we begin to make progress in defining the Semantic Web. He believes that the true battle is in coming to agreement on common definitions so that industries can begin to relate the right information together. This relating is what is required to move us from Level 1 to Level 2 in the evolution of data fidelity referred to in Figure 2.4. Only through standardization of tagging, as shown in Section 2.2.4.1, can we truly begin to interoperate together on progress to Level 2. It is these standards that are standards of language and definition that in the human sense allow us to communicate, interpret and understand. For computers to do this on a broad semantic scale the definitions must be far-reaching and precise. Despite the idea of the Semantic Web being known since 2001, this standard has yet to emerge on a broad scale, and only exists in small domains.

2.2.12 An Example Semantic Web Project: CS AKTiveSpace. One demonstration using Semantic Web is with the project known as CS AKTiveSpace [84]. CS AKTiveSpace is a Semantic Web application developed by the University of Southampton, UK which "attempts to provide an overview of current UK University-based research in Computer Science" [84]. Its application focused primarily on pulling non-tagged information from websites and then relating it together into ontologies in an automated fashion. Using the AKT Reverence Ontology [59] as a basis, it related content from 24,000 research projects from almost 2000 research facilities in the UK together in about ten million RDF triples. The application won the 2003 Semantic Web challenge [60], however it highlighted several shortfalls of the Semantic Web. With a relatively small input set they discovered issues with how best to sustain a metadata acquisition and harvesting activity. This included decisions about how best to model the harvested content, how to cope with the large number of duplicate items that are referring to the same objects or referents, and how this information is meant to be sustained and maintained. One might be impressed by the nearly 10 million RDF triples that were derived from the 24,000 research projects as being very thorough, however the research admits that "despite [their] successful harvesting of the web and exploitation of various organizations' RDF, [they] still [could not] reflect the richness of the queries [their] system [could] support because [they] do not always have enough content" [84].

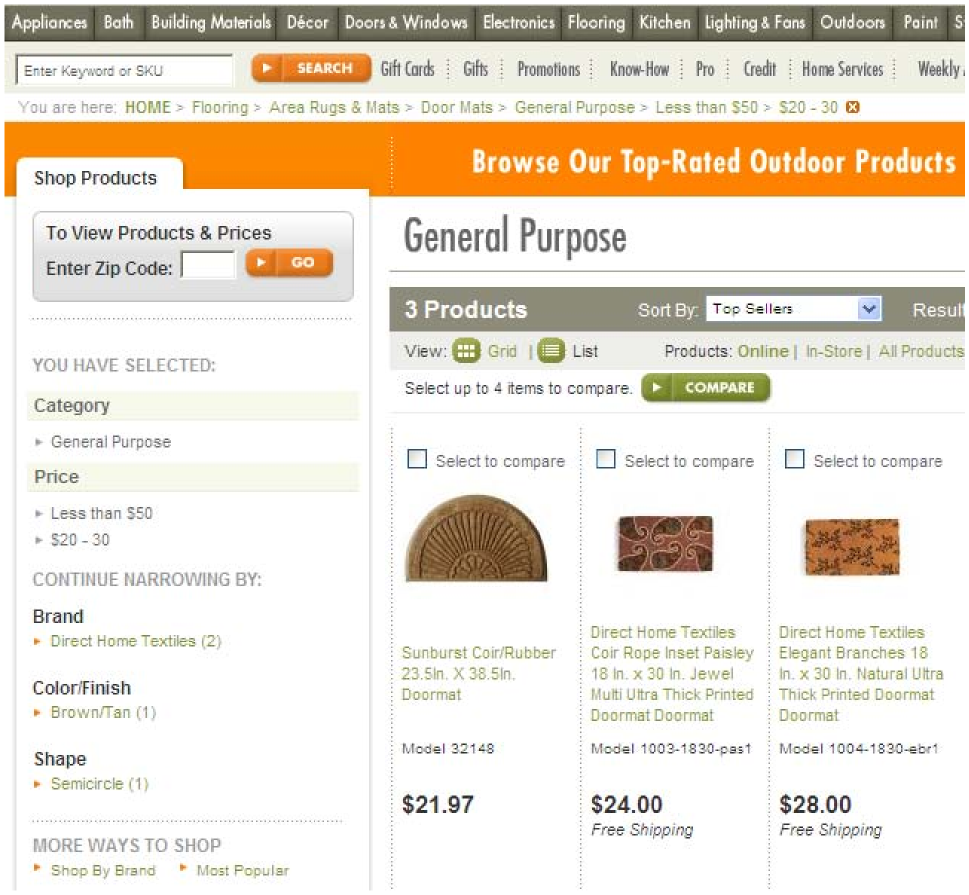
2.2.13 Semantic Web Applications. In Berners-Lee's initial vision for the Semantic Web, he envisioned computer applications that related the semantic information together in the place of humans. To this point, we have primarily just discussed the encoding of semantic information with web content and the methods for reasoning over that information. Ultimately the end goal is to move up the data continuum to knowledge. Having applications that can relate and merge data together to form information, and ultimately knowledge is the end goal of the Semantic Web.

Applications thus far have taken many different forms. Most are very tight in scope applying to the specific domains for which they were designed. As a result they tend to be highly specialized and therefore not very well suited outside the domain in which they were designed. We will discuss one type of application with the end goal to apply in a broad scale across many domains: the Semantic Web Browser.

2.2.13.1 Semantic Web browsers. Semantic Web browsers are web browsers that are tailored to take advantage of the semantic information that has been added to HTML documents. Many of the popular web portals and search engines today, like Google and Yahoo, take advantage of XML and RDF tagged information using ontology-based approaches to organize content on the Web [71]. The shortfall is that most of the content on the web is not thoroughly tagged. Thus their traditional keyword searches end up being more applicable to the web's current landscape. As a result they are more seen as web browsers that have been augmented by semantics, rather than true Semantic Web browsers.

Leveraging Semantic Tagging

There exists many websites that are taking advantage of semantic tagging. A recent trip to Home Depot online [48] noted semantic content being used. It allowed the products being viewed to be sorted by different category, price, and brand. As a result, an individual using the site can easily tailor their search dynamically. Figure 2.8 shows an example visit. You can see in Figure 2.8 that there are many choices that we use to segment products into different sections to include category, price, brand, color, shape, or popularity. Each is tailored by the type of product, thus suggesting a hierarchical structuring of products, and an XML/RDF method of tagging and retrieving data. While this additional organization is very welcome and dramatically improves the shopping and searching experience, it still falls short of what a true Semantic Web is capable of.



You are here: HOME > Flooring > Area Rugs & Mats > Door Mats > General Purpose > Less than $50 > $20 - 30

General Purpose

To View Products & Prices
Enter Zip Code:

3 Products

Products: Online | In-Store | All Products

Select up to 4 items to compare.

YOU HAVE SELECTED:

Category
- General Purpose

Price
- Less than $50
- $20 - 30

CONTINUE NARROWING BY:

Brand
- Direct Home Textiles (2)

Color/Finish
- Brown/Tan (1)

Shape
- Semicircle (1)

MORE WAYS TO SHOP

Select to compare	Select to compare	Select to compare
Sunburst Coir/Rubber 23.5In. X 38.5In. Doormat	Direct Home Textiles Coir Rope Inset Paisley 18 In. x 30 In. Jewel Multi Ultra Thick Printed Doormat Doormat	Direct Home Textiles Elegant Branches 18 In. x 30 In. Natural Ultra Thick Printed Doormat Doormat
Model 32148	Model 1003-1830-pas1	Model 1004-1830-ebr1
$21.97	$24.00	$28.00
	Free Shipping	Free Shipping

Figure 2.8: A Home Depot Online Buying Experience [2]

For example, consider if one wanted to compare products from another store to Home Depot's products. Assuming that the other online store had XML/RDF tagged information as well, it is easy to see that the data exists, but it needs to be assembled properly. An additional application would have to be brought in to merge the two data sets together. This is because Home Depot's web server is fixing the presentation of their website in a certain manner, while when we bring in another website and attempt to aggregate the information, we are no longer just viewing a page generated within the Home Depot server's scope of control. Thus, the home depot server and its web site acts like a portal that is in full control of the content provider, and only gives presentation control to the end user through certain controls that are limited by the scope of the web site developer. A true Semantic Web browser is capable of aggregating content from several sources in a way that is not limited by what a web site developer *thinks* an end user will want, it actually gives full control to the end user.

Mashups

Another example of Semantic Web Browsing exists in mashups. A mashup is defined as "a web application that combines data and/or functionality from more than one source" [36]. Mashups are becoming very common. One website that lists and tracks websites is *www.programmableweb.com.* Here people list their mashup sites that they have created. You can find everything from messaging centers that combine data from cell phone messaging sites, with home value sites, to mapping sites that combine data from listing sites like *craigslist.com* [11]. Figures 2.9 and 2.10 show how the mashup sites are broken up by category and how their quantity has grown recently. Again, similar to the Home Depot site discussed earlier, these mashup sites can be very informative, but are hard-coded solutions from web developers with the same inflexibilities unless you bring computing programming skills and develop your own sites for your own purposes.

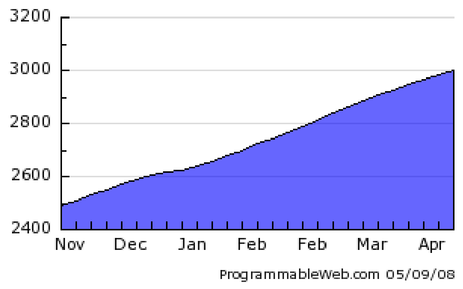


Figure 2.9: Mashups over time [44]

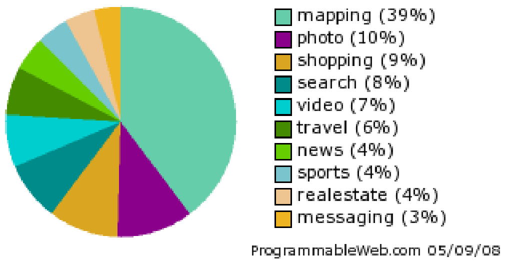


Figure 2.10: Breakout of Mashups on Web [36]

If mashups are the style of information that a developer is looking for, but they are not that savvy in writing the code for it, MIT developed a tool called Exhibit that was acquired by Google on Mar 27, 2008 [20]. The tool is a three-tier web application framework written in Javascript. An individual that only wants to show a few hundred records of data on maps, timelines, scatter plots, interactive tables, etc., without having to learn SQL, ASP, PHP, CGI, or any other language can use Exhibit to do it for them. All they need to do is to write a simple data file, and an HTML file to specify how the data should be shown. The developer writes the data and the presentation; Exhibit does the rest [49].

Hakia

There are a few Semantic Web searches that have been deployed on the Internet. One that uses true semantics is *www.hakia.com*. Hakia is a semantic search engine that brings relevant results based on "concept" match rather than "keyword" match or popularity ranking as current search engines use [28]. It uses an algorithm it calls "SemanticRank" that is comprised of solutions from the disciplines of Ontological Semantics, Fuzzy Logic, Computational Linguistics, and Mathematics [28]. It has built a series of ontologies to pull out the meaning of the sentence structure in documents listed on the web and extracts all the possible queries that could lead to this sentence [27]. Hakia is currently in Beta testing and is using 2008 to harvest results for their Hakia development [28].

PowerSet

Another Semantic Web search that exists on the internet is a search called PowerSet at *www.powerset.com*. PowerSet's first product is a search and discovery experience for Wikipedia, that was launched in May 2008 [3]. PowerSet differs from Hakia in that it does not use an ontology as its basis for semantic search, it rather uses a system that parses the syntax of the sentence and guesses matches based on statistics. This approach means that for questions that do not match previously encountered syntactical patterns, the system will not be able to find answers, even if there are in fact answers in the database [37].

Faceted Browser

Lastly, there exists a type of tool that is known as a "faceted browser." A "faceted browser" is a software tool that uses faceted navigation to help users browse information. "Faceted navigation" gives the users the ability to find items based on more than one dimension, to see breakdowns and projections of the items along different axis, which helps users gather insights about the data they are exploring [21]. Faceted browsers are especially powerful when they are used over a dataset that is fully tagged with RDF or OWL. That is because faceted navigation is context dependent, meaning that available facets, facet values and their count are based on the current set of results that the user is browsing. In large datasets this constant recalculating of facets can become time intensive, which is handled by caching and pre-loading the most common combination of searches. Tools that are faceted browsers include Longwell [34], Flamenco [24], Apache Solr [55], Endeca [18], and WorldCat [58].

2.2.13.2 Reasoning Engine Tools. There are several tools that have been built over the years that are implementations of reasoning engines as defined in Section 2.1.5. We will discuss the Closed World Model, Jena, Racer, Pellet, and Jess.

Closed World Model

The Closed World Model (CWM) inference engine was written in Python by Tim Berners-Lee and Dan Connolly. It is a general-purpose data processor for the Semantic Web. It stores RDF triples in a query-able triples database and performs inferences as a forward chaining first-order predicate logic inference engine. It can be used for querying, checking, transforming, and filtering information [64] [12].

Jena

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena is open source and grown out of work with the HP Labs Semantic Web Programme [32]. Jena is the most popular reasoning engine according to a survey done in late 2007. In the survey it reported that Jena was used by 53.6% of individuals surveyed [68]. It is clear that the fact that Jena was built on Java, is open source, works with several different schemas and formats, and is easily extendable are key reasons for its high usage.

Racer

Racer was the # 2 most used reasoning engine according to a survey done in 2007 stating that it was currently being used by 28.2% of individuals surveyed [68]. Racer and its derivatives RacerPro, RacerPlus, RacerMaster, and RacerPorter are developed by Prof. Dr. Volker Haarslev, Kay Hidde, Prof. Dr. Ralf Moller and Michael Wessel who in September 2004 founded the Racer Systems GmbH & Co. KG to give the RacerPro software a home in the commercial business world [46]. Racer sits on top of OWL-DL, the most commonly implemented version of OWL. Figure 2.11 shows how Racer works with external applications, databases, and other semantic languages.

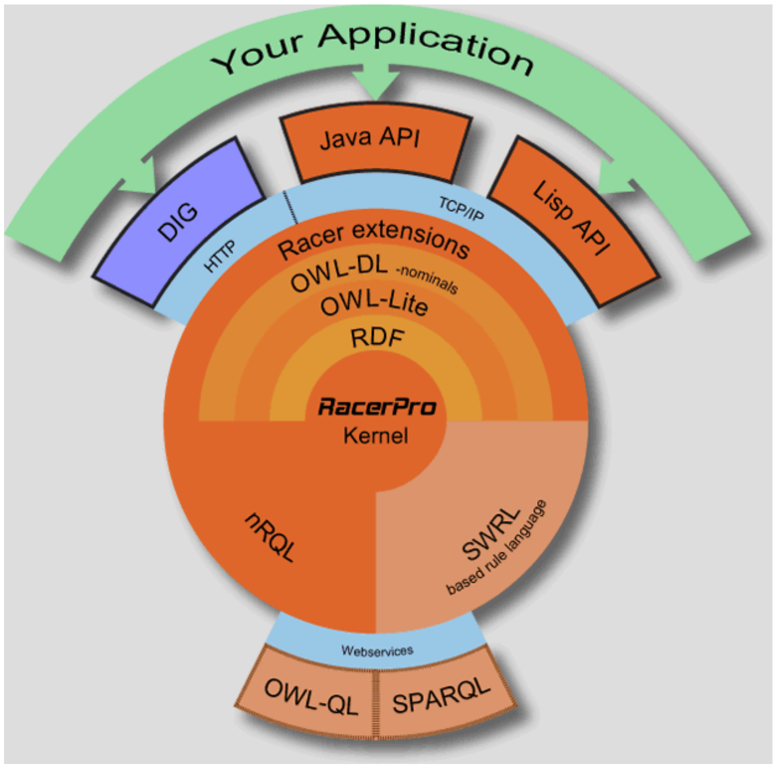


Figure 2.11: RacerPro application interface layers [46]

Pellet

Pellet is an open source, OWL-DL reasoner in Java that is developed, and commercially supported, by Clark & Parsia LLC. Based on the tableau algorithms for expressive Description Logics (DL), Pellet supports the full expressivity of OWL-DL, including reasoning about enumerated classes. Pellet supports all the features proposed in OWL 1.1, with the exception of n-ary datatypes [42].

Pellet reasoner can be set up to work in Jena without using the DL Implementors Group (DIG) interface. This allows for improved speed and overcomes some of the limitations in the DIG protocol [32]. Pellet was also mentioned in the 2007 survey as the third most widely used reasoning engine with 23.4% of individuals surveyed saying they have used it [68].

Jess

Jess [33] is a rule engine and scripting environment built on Java. It uses an enhanced version of the Rete algorithm [72] to forward or backward chain based on a set of facts and a set of rules in order to discover new facts. Rete is efficient at solving difficult many-to-many matching problems inherent in deduction. While Jess is written in Java, and can be called in from Java software code, it also is written as a scripting language to allow it to take input and produce outputs from many different sources. The developers have also created plug-ins to development tools such as Eclipse [17].

Jess is the reasoning engine that was chosen to be integrated into our system to generate knowledge from vehicle tracks. It was chosen primarily because of its simplilcity to integrate into Java and Eclipse. Its Rete engine is powerful in deduction, and it was easily scalable for large and small rule sets. It is also portable where the rules used in a Java code version can be reused exactly if it were adapted to integrate with other software in its scripting form.

2.2.14 Summary of Semantic Web. Throughout the discussion on Semantic Web there are a few key take-aways that should be restated.

1. We must have richly tagged data at the core in order to extract it. This should be provided by the publisher, or we must design in a method to extract it from untagged data.
2. Ontologies are the intersection and translation point of the real world semantics to computer language semantics. Only with tight, sound ontologies can we hope to integrate and relate disparate data together into information. OWL, built on RDF, XML, and RDFS, has been largely the language of choice for ontologies in the past few years.
3. Having richly tagged data and tightly built ontologies are useless unless we have the means to make use of that data through a semantic agent, semantic browser, faceted browser, or a reasoning engine.

2.3 Computer Video Indexing

One key area of knowledge generation from surveillance video is the processing of video into content that is understandable by a computer. Video must first be processed and tagged in a computer in order for a computer to be able to work on it, understand it, and reason over it. The last few decades have had substantial progress in the area of storage technology. Individuals are able to processes large amounts of data for themselves personally. That has brought significantly more multimedia in the form of pictures and video content on their computers and available on the web. The web has grown as well, but most of the search capabilities are focused on text-search, because it is much easier for algorithms to process text-based information instead of multimedia information. That is because multimedia data needs many stages of pre-processing to be indexed in such a way that it is relevant to run queries against. Current day research has dealt with the indexing shortfall somewhat, but still continues to have areas with shortfalls in capabilities. Most recently research has focused on the use of internal features of images and videos computed in an automated or semi-automated way. Semi-automated processing adds the assistance of human interaction to automated processing. Automated analysis uses statistics to approximate the content features and determine if the features match a set of pre-loaded features. Processing has gradually built from solely tagging items syntactically to identifying semantic features. These systems, known as Content-Based Retrieval (CBR) or Content-Based Information Retrieval (CBIR) systems, operate at a semantic level and use motion-features in addition to other features like color, and object identification to build a greater awareness of what is going on in the image or video. CBR systems use various techniques to build semantic information from images. We will discuss some of the these techniques including abstracting multimedia content, matching techniques, and the application learning methods to multimedia [79].

2.3.1 Abstracting Multimedia Content. Multimedia content can be modeled as a hierarchy of abstractions. At the lowest level are individual pixels consisting of basic information such as color or brightness. At the intermediate level are objects and their attributes formed from the individual pixels. Above the intermediate level is the human level concepts that interpret the objects interactions together. Using these levels of abstraction, descriptive features can be categorized as either syntactic features or semantic features. Syntactic features are low-level characteristics such as an object boundary or a color histogram. A semantic feature represents an abstract feature such as the label *road* assigned to a region of an image.

2.3.1.1 Syntactic Indexing. Several CBR systems use syntactic features as the basis for matching, and then use an interface via Query-by-example or Query-through-dialog-box to retrieve images. Query-through-dialog-box was the first method that was used and it has limitations that the end user must have intricate knowledge of how the programmer processed and tagged the data in order to put together meaningful queries with accurate results. Query-by-example is a method where the user is presented a number of example images, and the user indicates which image is the closest to their desired search. Then the system analyzes that (or those) image(s) for color, shape, texture, spatial distribution, and features to arrive at a new list of candidates [79]. This type of searching has had moderate success, but suffers from a lack of semantic understanding. While a user can help in the feedback loop, the computer still is uncertain which particular features the user finds most important. If a picture of a bird, in front of a waterfall, in the middle of the jungle is chosen, and the computer accurately pulls out those three features, how does it know which feature was the driving reason for the image being chosen? It could be one of those features, a combination of those features, or even the presence of those features without an absent feature (for example, without a rainbow). It could also be the fact that the bird was blue, or the jungle had a certain plant in it. It is very difficult for the computer to reach this level of understanding, especially the absent feature making the Query-by-example method sometimes frustrating for users.

2.3.1.2 Semantic Indexing. Building on syntactic indexing, some research has been done in the direction of developing techniques based on analyzing the content of images at a higher level. The key benefits that this approach gives us are: 1) higher-similarity ratings are produced by more relevant semantic features as opposed to syntactic features, and 2) the performance of search is improved by using semantic cues as compared to just looking at low-level features [79]. Some work in [89] builds an *a priori* model of a video's structure based on domain knowledge. This is done often with newscasts or sports broadcasts as they sometimes have parsable information that is presented in a consistent repeatable form. This work has shown that problem domain knowledge can be used to simplify the processing of semantic information in video. Other work has used the pacing of video shots or cinematic modeling to semantically tag video data and classify it into one of many different categories of video [74] [83]. The pacing of video shots involves how quickly video cuts to different sections (faster for action, slower for romance) while cinematic modeling deals more with how a video shot is set up (i.e. camera zoom out for end of a show, or zoom in on a character while they're thinking). This work has further demonstrated in different problem domains the usefulness of bringing problem domain information into the processing of the data for increased benefit.

2.3.1.3 Television Related Data Processing Efficiencies. Most the research done has been in the area of indexing television shows to include various sports and news broadcasts. In these problem domain areas there exists some commonalities that are used to extract semantic information. Examples of these include indexing shots made in a basketball game which are related to changes of score on the screen in a caption section of the screen. Other important elements of a sports game can be related to slow-motion replays. It is important to note that the better the data is understood, the more accurate a processing system can be made that indexes it properly [79].

Additional work has been done on multiple sporting events with the goal of creating a system that works for several different domains. Ultimately they still focus on television related data processing which has its underlying assumptions related to broadcasting sports, broadcasting news, or cinematic visual effects. These assumptions are good within their particular domains, but may not apply well outside their domains. An example of this would be applying indexing tailored towards basketball games on ESPN to a different data set such as high altitude surveillance video. Many video indexing hierarchies are built with a particular domain in mind, and the semantic information made from assumptions that hold within that problem domain. What makes them useful in these problem domains is the assumptions and the information that can be extracted as a result. Unfortunately these assumptions may not hold in different problem domains and that makes them less useful outside of the specific problem domain in which they are developed.

2.3.2 Matching Techniques. Matching techniques are a method of finding similarity between two sets of multimedia data (videos or images). It focuses on using parameters like the level of abstraction of features, distance measures, and weighting features to find matches to a reference. In VisualSEEK [85], a query is specified by the colors, sizes and arbitrary spatial layouts of the color regions with their absolute and relative spatial locations. It generates a value of a function considering all those elements together of a reference area of an image. It then computes that function for all images known and returns those images that are closest in value to that function [79]. Key to this approach is how the function is computed and what factors and features it uses to create its function values.

There are some shortfalls in the approaches that have been used in matching techniques. Many use Euclidean distance of low-level features without any method to automatically generate the weights of the features. They also assume a Gaussian probability distribution of the features which may or may not be appropriate for the given problem domain.

2.3.3 Learning Methods Applied to Multimedia. There have also been strategies of using learning algorithms on representative training data to help construct a model of a particular domain. Methods such as K-Nearest Neighbor classifiers [88], decision trees with positive and negative examples [70], linear discriminant analysis [86], and Markovian frameworks [81] have been applied to multimedia data with positive results. The difficulty with all learning methods is to have the right breadth and depth of learning in order to accurately identify what items are of interest.

2.4 Geographical Information Systems and Databases

Geographical Information Systems (GIS), also called Geographic Information Systems are systems that "integrate hardware, software, and data for capturing, managing, analyzing, and displaying all forms of geographically referenced information" [56]. They make visualizing and analyzing geographical information simpler. GISs in their simplest form are databases that have tables to store data that represent various layers of information with respect to a certain geographical area. These layers can represent physical items like roads, a sewer network, water supply lines, electrical lines, or buildings by using the basic geometries (shapes) of point, line, and polygon. They also can represent images or video from above of a location. All these layers are related to one another by a common coordinate system like Latitude/Longitude or Universal Transverse Mercator (UTM). Once layers are related in a common coordinate system they can related one to another in space both visually and with queries (i.e. where sewers intersect with roads, or what buildings are within 300 meters of water).

2.4.1 ArcGIS. ArcGIS is a compilation of software produced by the Environmental Systems Research Institute, Inc. (ESRI) that together constitute a GIS. Under the overarching ArcGIS name, there are versions that are tailored for individual use (ArcGIS Desktop) and multi-user use (ArcGIS Server) [19]. ArcGIS Desktop provides all the basic functionality of a GIS including importing data, drawing shapes manually, querying data spatially, and viewing the data (shown in Figure 2.12).

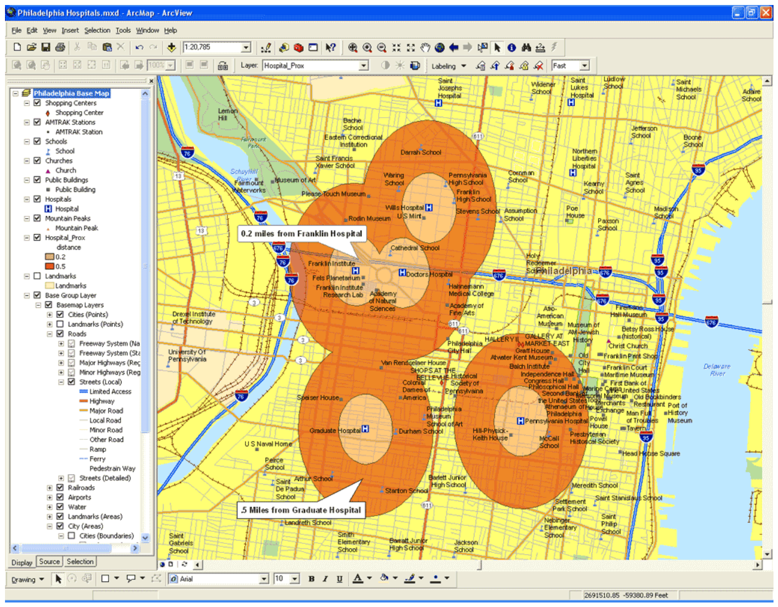
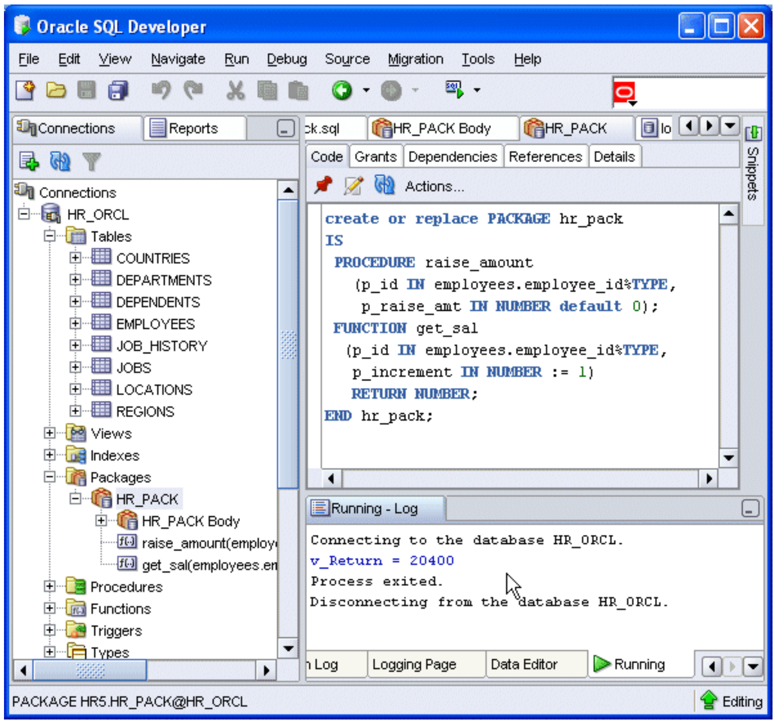

Philadelphia Hospitals.mxd - ArcMap - ArcView
0.2 miles from Franklin Hospital
.5 Miles from Graduate Hospital
Philadelphia Base Map
Hospital_Prox

Figure 2.12: An example ArcGIS screenshot showing the road network and distances from hospitals in the area [5]

2.4.2 Oracle Spatial. Oracle Spatial [40] is a subcomponent to the Oracle 11g release by the Oracle Corporation. It is a more database-centric approach to relating spatial data. Within the Oracle Spatial subcomponent, Oracle provides an extension to database structured query language (SQL) that allows users to query the database using spatial functions. An example of these spatial functions might include finding all Starbucks in a five mile radius, or within a zip code, sorted by distance. It might also include finding which roads intersect with other roads and the points at which they do. An additional benefit to Oracle Spatial is that SQL can be called to the database from Java code and manipulated in Java. Oracle has also provided a tool "Oracle SQL Developer" (screenshot in Figure 2.13) that can be used to test and debug SQL code prior to calling it from Java.

2.4.3 GIS data resources. Essential to getting the most benefit from GIS systems is the ability to find data of an area of interest. While typical GIS systems



```
create or replace PACKAGE hr_pack
IS
 PROCEDURE raise_amount
    (p_id IN employees.employee_id%TYPE,
     p_raise_amt IN NUMBER default 0);
 FUNCTION get_sal
   (p_id IN employees.employee_id%TYPE,
    p_increment IN NUMBER := 1)
    RETURN NUMBER;
END hr_pack;
```

```
Connecting to the database HR_ORCL.
v_Return = 20400
Process exited.
Disconnecting from the database HR_ORCL.
```

Figure 2.13: An Oracle SQL Developer screenshot [35]

have the capability to manually draw shapes or lines, this work can be very tedious and time consuming if done for a large area. It is very important that the information is drawn or loaded correctly, because incorrect data leads to incorrect results when the data is related and queried. Fortunately there are companies that have imported and done much of the work for the populated areas of the world. ESRI sells a Data and Maps Media Kit [5], which is updated annually that contains more than 24 GB of data including:

- Basemap and thematic MXDs for Canada, Europe, Mexico, the United States, and the world
- Commercial data from Tele Atlas, AND Mapping, DMTI Spatial, WorldSat, EarthSat, EuroGeographics, Michael Bauer Research, World Wildlife Fund, SIGSA, and ESRI
- Ninety-meter Shuttle Radar Topography Mission (SRTM) dataset
- All levels of U.S. Census geography and ZIP Codes

- TIGER 2000-based StreetMap USA data

There are several other web sites that have GIS data available to download. One maintained by Stanford University [6] lists thousands of websites sorted by areas of interest (climate, hydrography, topological) and area of the world (Middle East, Afghanistan, US, Europe, etc.). Garmin [7], a maker of car, boat, and personal GPS navigation devices, also has data that can be downloaded, and 3rd party affiliates that have data in other areas of the world. In total, there are many resources for finding data available to use. For areas that have no data, resources would have to be expended to create the data in order for a GIS-based solution to be beneficial.

2.5 Surveillance Domain Knowledge

Today's security forces around the globe focus strongly on deterring, detecting, and preventing terrorist acts. While terrorists have a benefit that they do not fit within the traditional nation-state order of the world and often hide across nation-state boundaries, they also have disadvantages. The main disadvantage that terrorists have in conducting their operations is that they are extremely dependent on surveillance done during target selection. This surveillance is done as part of target selection during a six-stage attack cycle: target selection, planning, deployment, attack, escape, and exploitation [53]. Even before a plan is put together, terrorist are out in the open, watching potential targets to understand the behavior of targets and assess their weaknesses. Only when targets are fully understood can a plan be put together to exploit the weaknesses. This surveillance often takes several weeks to execute exposing the terrorists to detection and interdiction well before an actual attack is to take place. Detecting surveillance is one of the best ways to counter terrorist acts. Countersurveillance [53] is the proactive means of spotting terrorist and criminal surveillance during the target selection and planning stage. By using countersurveillance, security personnel charged with protecting targets can learn about the individuals which are conducting surveillance and perhaps interdict them and break up the group before the planning is complete. A system that would help accomplish countersurveillance by using automated knowledge generation techniques would be invaluable to security personnel.

2.5.1 Suspicious Activities. One method that is useful in detecting surveillance is being able to detect suspicious activities. In a June 2008 release to the public [30], the Department of Homeland Security (DHS) lists the following examples of suspicious activities:

- Multiple sightings of the same suspicious person, vehicle, or activity, separated by time, distance, or direction.
- Individuals who stay at bus or train stops for extended periods while buses and trains come and go.
- Individuals who carry on long conversations on pay or cellular telephones.
- Individuals who order food at a restaurant and leave before the food arrives or who order without eating.
- Joggers who stand and stretch for an inordinate amount of time.
- Individuals sitting in a parked car for an extended period of time.
- Individuals who do not fit into the surrounding environment because they are wearing improper attire for the location or season.
- Individuals drawing pictures or taking notes in an area not normally of interest to a tourist or showing unusual interest in or photographing security cameras, guard locations, or watching security reaction drills and procedures.
- Individuals who exhibit suspicious behavior, such as staring or quickly looking away from individuals or vehicles as they enter or leave facilities or parking areas.

DHS also list other activities which should cause a heightened sense of suspicion as [30]:

- Suspicious or unusual interest

- Surveillance (suspicious in nature)
- Inappropriate photographs or videos
- Note-taking
- Drawing of diagrams
- Annotating maps
- Using binoculars or night vision devices

Police departments across the United States also list other items as suspicious activities [8] [9] [47]. These include seeing a car driving around the block several times, individuals watching houses or kids, peeking inside windows of houses, and going door to door knocking at front doors and back doors. Any of these events when taken individually may not account for much, but if they are related together they add credibility to the possibility of surveillance taking place.

2.5.2 Use of Social Networks in Surveillance Activities and Counterterrorism.

Social Networks are a graphical representation of how individuals interact with one another. They are often used to pictorially represent the relationships among people within an organization in order to better understand how the organization operates. An example social graph representing a social network is shown in Figure 2.14. Related to surveillance activities, one goal is to understand who the actors are in an organization and where they operate. When an individual has been identified as suspicious they are put under surveillance by friendly forces in order to better understand who the suspicious individual is operating with and where their illegal activities might be taking place. The surveillance begins forming the links of which individuals interact with the suspicious individual and which locations the suspicious individual visits. The more links that are populated from the suspicious individuals the more we begin to understand what activities are going on and where they are happening. This understanding leads to the construction of plans on how best to disrupt the suspicious individual and their plan by disrupting the organization (individuals), disrupting their communication (links), or disrupting their locations of operation.

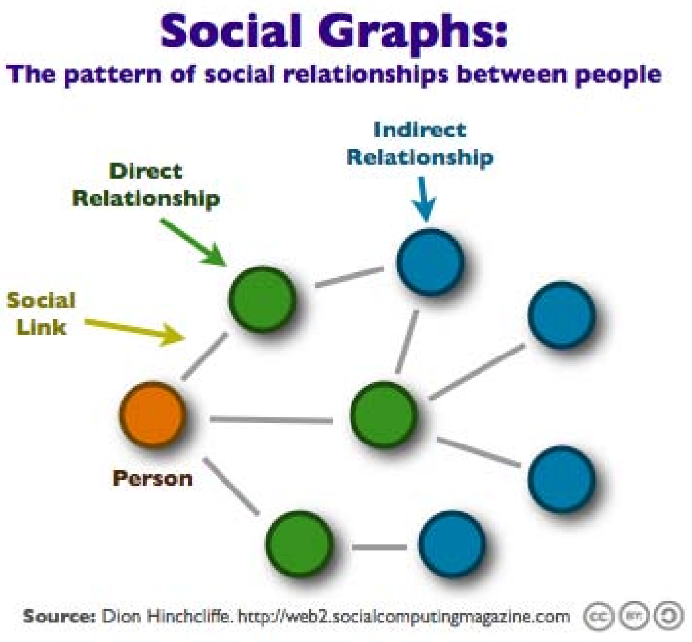

Social Graphs:
The pattern of social relationships between people
Indirect Relationship
Direct Relationship
Social Link
Person
Source: Dion Hinchcliffe. http://web2.socialcomputingmagazine.com

Figure 2.14: An example social graph [14]

In counterterrorism and counterinsurgency, social networks provide a means of understanding a terrorist group, who the major actors are, and how best to disrupt their operations [61]. Specific successes of applying social networks to counterinsurgency and terrorism include the search and capture of Saddam Hussein [50] and the discovery and arrest of a Canadian terrorist cell in 2006 [10]. The National Security Agency (NSA) keeps phone logs and monitors internet chat rooms in order to apply social network analysis to pre-empt potential terrorist attacks [31]. While social networks are a beneficial approach to counterterrorism, they come at a cost. Some argue that the NSA's use of monitoring traffic is a breach of privacy under the Constitution [31]. In addition, many resources are expended to included personnel, and intelligence collection capabilities to monitor suspicious individuals. Another factor is the time it takes to collect the information. Just as it takes weeks and sometimes years for terrorists to collect information about potential targets, it often takes just as long for friendly forces to collect enough information to act credibly against enemies. Any improvement that would decrease the time or resources required to build a social network would be beneficial to finding and interdicting terrorist activities.

2.6 Summary

In this chapter, we discussed the background of AI, Semantic Web, image processing, Geographical Information Systems, and basic surveillance tactics as it relates to knowledge generation. To automatically generate new knowledge, we will build a system using the Semantic Web concept of moving up the data fidelity ladder (See Figure 2.4). To accomplish this, we will use an inference engine (reasoning engine) described in Section 2.1.5 to generate new facts from existing facts in accordance with a series of rules derived from information on suspicious activities (Section 2.5.1) in the surveillance domain. We will use the reasoning engine tool Jess (discussed in Section 2.2.13.2) as our reasoner. We will use ArcGIS (Section 2.4.1) to help us visualize results and Oracle Spatial (Section 2.4.2) to help us integrate vehicle tracking information with static GIS information, and also integrate it with the Jess reasoner. We feel confident that this approach will provide some benefit because of the success that others have had processing images with learning methods (Section 2.3.3). The end goal is to integrate a Jess reasoner loaded with suspicious activity focused rules with vehicle tracks, geographical information and the tools to relate them all together (in Oracle Spatial) in order to automatically generate new knowledge from vehicle tracks and GIS layers.

III. Casing Event Detection Methodology

Over the past several years, interest in applying automated computer processes to video data has grown. The growth in computer storage space along with the growth of digital imaging components has created a large repository of digital images and video. Surveillance applications using these large repositories provide capabilities beyond the typical rewind and playback used prior to computer-centric video processing [89]. With the advent of these large video sources, a need for automated systems to process the large amount of information has developed. These automated systems are helpful in some applications, but only to the degree that they accurately detect objects and semantically interpret them the way a human would. To make automated systems useful, the systems must detect objects, and potentially identify relationships between the objects to detect events.

One event that would be of interest to security personnel using persistent surveillance is the detection of terrorist pre-attack surveillance activities. This pre-attack surveillance portion of a terrorist's execution cycle is the most vulnerable to counterterrorist detection and has the greatest probability of disrupting a terrorist event before it occurs [53]. The suspicious activity that we detect is a vehicle that is watching a certain location. We will call this activity a "casing event" or "casing."

Figure 3.1 shows the traditional flow of processing in a visual surveillance system. Throughout the process, the focus is on abstracting pixel data provided by the images to more abstract concepts such as objects (cars, person, package, building), then using motion detection to track those objects and further confirm the detection. The end goal is to model the behavior of these objects and determine the activities that the objects are involved in to assist in the detection of events that humans are interested in. These events are logged, and their related images stored into a database for retrieval.

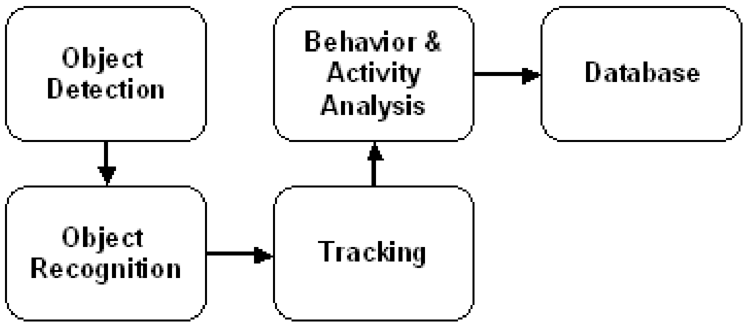


Figure 3.1: Traditional flow of processing in a visual surveillance system [87]

3.1 Overview

The overall paradigm used to approach knowledge generation from persistent video is applying the concept of moving up a pyramid of data fidelity [69] from basic "things" to "worlds" (see Fig. 3.2). At the base of the pyramid, "things" relate to syntactic detections of vehicle movements. We used further syntactic processing to turn those detected vehicle movements into detections of whether or not a vehicle has turned left or right in its driving path. This begins crossing into the portion of "knowledge about things" in the model. An algorithm is built that processed the detection of turns deterministically. Then, the track location waypoints that the track travels on will be populated with semantic information related to the contextual interactions of each waypoint with other GIS layers. The turn detections, combined with the point semantics iss reasoned over by an artificial intelligence reasoning engine to detect more complex events like an event where an adversary was driving circles around a site suspiciously watching that site (i.e. casing). This is discussed in Section 3.4.

3.2 Turn Detection

Initially, an algorithm for the detection of turns will be designed and tested. The results of the algorithm will then become the basis of facts for a reasoning engine to process higher level event detection. The specific details of the turn detection methodology and additional results are provided in Appendix A.

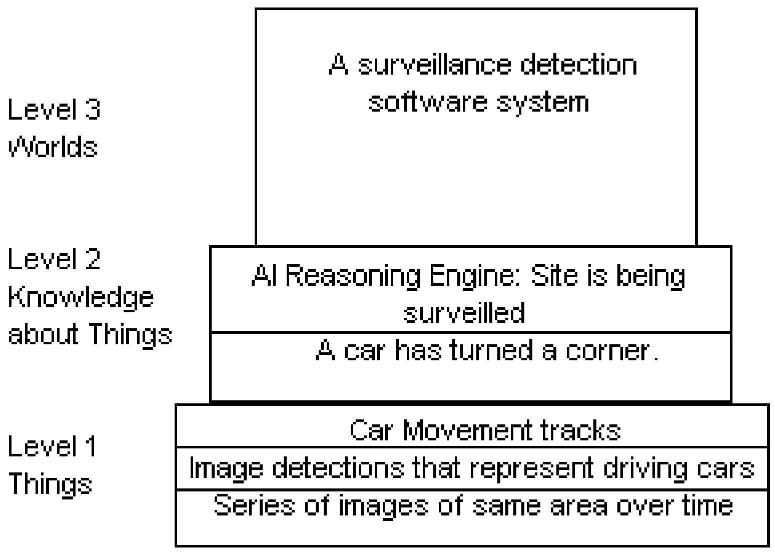

A surveillance detection software system
Level 3 Worlds
Level 2 Knowledge about Things
AI Reasoning Engine: Site is being surveilled
A car has turned a corner.
Level 1 Things
Car Movement tracks
Image detections that represent driving cars
Series of images of same area over time

Figure 3.2: Evolution of data fidelity as it relates to Persistent Surveillance Video

3.2.1 Problem Definition. The first element in detecting a casing event is to detect when a vehicle track has changed course as a result of a deliberate action (i.e. the vehicle has turned). Detecting the points that are turning provides a basis that we use to move up the data fidelity ladder.

3.2.2 Workload. The workload used is persistent video data over an urban campus area as shown in Figure 3.3. The video is post-processed to detect moving vehicles denoted as tracks (vehicular tracking in and of itself is a separate research effort outside the scope of this thesis [73]). Tracks are listed in a common coordinate system to relate to other static information from a Geographic Information System (GIS). Global Positioning System (GPS) track data from the same area was used to represent vehicular track input to our model. The GPS data [62] is a good representative of the persistent video data because GPS data logs position coordinates at nearly the same time interval that typical persistent video systems do. Both sets of data are also not without fault, as both GPS data and tracks generated from persistent video can have areas of missing detections caused by system errors.

Figure 3.3: A sample overhead surveillance shot [63]

Since turn detection is a subset of the overall events we intend to detect and model, we will only run tests on a portion of the 140,000 detections for turn detection analysis. These detections represent 14 different vehicle tracks and a total of 25.5 hours of driving time. Specifically, in turn detection we will run it against 3 of the 14 vehicles, representing 106 minutes of total driving time. This will leave data sets available for future testing against larger events.

3.2.3 Results. The results of testing on the turn detections algorithm show that the following values for parameters works the best for an urban area. When asking if a particular point is turning we should look behind that point at least 45 meters, ahead at least 20 meters, and use a 20 degree turn angle, and 0.5 mph speed differential. In this configuration, we can say, with 95% confidence, that the turn detection system detects 81.4% to 92.94% of turns, and given the system detects a turn (P(Real Turn—Detected Turn)) we can say, with 95% confidence, that it is a real detection between 84.34% and 95.5% of the time. The detection of these turns will assist us in determining whether a casing event has occurred which will be discussed in more detail in Section 3.4. Further detail on the definitions of these parameters and the extended results of turn detection can be found in Appendix A.

3.3 Point Related Semantics

In addition to using output from a Turn Detection System (TDS) additional context can be added to tracks to further move up the data fidelity pyramid. By relating a track to objects that it is on or near, additional context is created which can be used to improve accuracy to detecting higher level events.

3.3.1 Problem Definition. Continuing with vehicle tracks that have been augmented by turn detection, it is necessary to distinguish between turns that occur on streets, in parking lots, in intersections, from streets to parking lots and vice versa. This provides added context to determine if a track may just be beginning or ending, and whether it is driving in a normal or abnormal manner.

3.3.1.1 Goals and Hypothesis. The goal of point related semantics to add context to vehicle tracks by relating it spatially to other information like the proximity of roads, parking lots, and buildings. Oracle Spatial functions that were reviewed in Section 2.4.2 will be used to further populate semantics on the GPS track data.

3.3.2 Workload. The workload that will be used in the point related semantics is the same workload that is used in turn detection. We will, however use all 14 vehicles and tag semantics to all vehicles, not just three vehicles as was used in turn detection. In addition to the tracks, the results of the turn detection will also be brought in, as well as other layers that represent the road network, parking lots, and buildings in the area.

3.3.3 Results. The details of processing the GPS way points to include the SQL code are discussed in detail in Appendix B. The result is that each GPS way point is updated with the additional context of whether that way point is turning left or right (from Turn Detection), is in a parking lot, is on a road, and is in an intersection. A screen shot of these results as plotted in ArcGIS is shown in Figure 3.4. Referring to Figure 3.4, the track driving around the OSU campus is highlighted yellow when it is in an intersection, red when it is turning right, and green when it is turning left. Tagging related to parking lots and on roads is omitted to provide a clearer picture. With these additional semantics, it is now possible to determine when a vehicle is turning while in a parking lot ($parking$ = TRUE, $turn$ = TRUE), turning in an intersection ($intersection$ = TRUE, $turn$ = TRUE), going straight through an intersection ($intersection$ = TRUE, $turn$ = FALSE), or turning when not in an intersection ($intersection$ = FALSE, $turn$ = TRUE). These insights allow us to model behavior more accurately as turning into a parking lot from a road, means something different than turning in an intersection, as an example. The semantics augment the turn detection to give better meaning to the behavior of a track allowing us to ask higher level questions with more context.

3.4 Detecting Building Surveillance Activities

One event that would be of interest to security personnel using persistent surveillance would be the detection of a terrorist pre-attack surveillance activity. Discussed in Section 2.5.1 were several suspicious activities that terrorists accomplish during their pre-attack surveillance of a potential target. As Section 2.5 discussed, this pre-attack surveillance portion of a terrorist's execution cycle is the most vulnerable to detection and has the greatest probability of stopping a terrorist event before it occurs. The suspicious activity that we will try and detect is a vehicle that is looping around the block as it is watching a certain location. We will call this looping around the block in a vehicle a "casing event" or "casing."

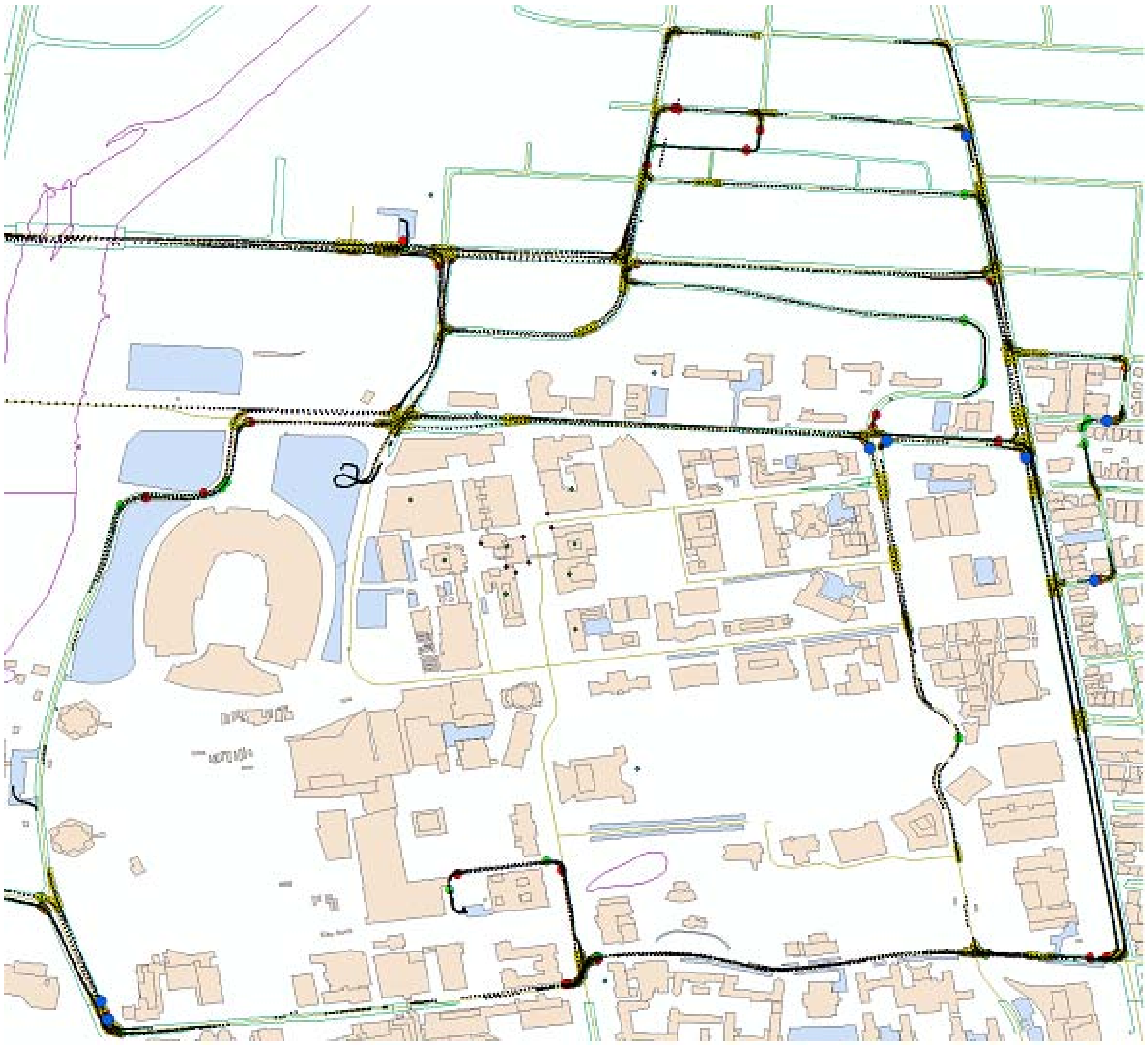


Figure 3.4: Pictorial results of semantic interpretation of GPS tracks

3.4.1 Problem Definition. The problem is to detect possible casing events amongst the track data and have high enough confidence in the detection to avoid a false positive.

3.4.1.1 Goals and Hypothesis. The goal of detecting a casing event is to detect every casing event that occurs and to report meaningful information that can be used to judge whether a detection is accurate or not. The hypothesis is that the algorithm will be able to detect casing events by tracking the turn behavior of a vehicle and reporting the possible locations that are being cased.

3.4.1.2 Approach. The approach is to use overhead persistent surveillance video augmented by turn detection and point based semantics to detect casing events. GPS data will be used to represent the results of tracking vehicles by overhead persistent surveillance which will be processed with turn detection and point semantics, and then fed into a Jess reasoner (See Section 2.2.13.2) to detect casing events.

Finite Automata Model for Casing Event

The Jess reasoner is used to build a finite automata as shown in Figure 3.5 to detect a casing event. Each input of a Left turn (L) or right turn (R) will transition from one state to another state. There are some instances where multiple transitions will occur simultaneously. For example, if the current state is "3 Right", and a Left turn (L) occurs, then the Jess reasoner will transition to both the "2 Right" and "1 Left" state. The Jess reasoner is able to catch all casing events in one pass through the GPS data and catch such events that are similar to having three right turns, a Left turn, followed by three more right turns (and all combinations that still constitute a full 360 degree completed loop).

3.4.2 System Boundaries. The system that is being developed is called the Casing Event Detect System (CEDS). A block diagram of the system is shown in

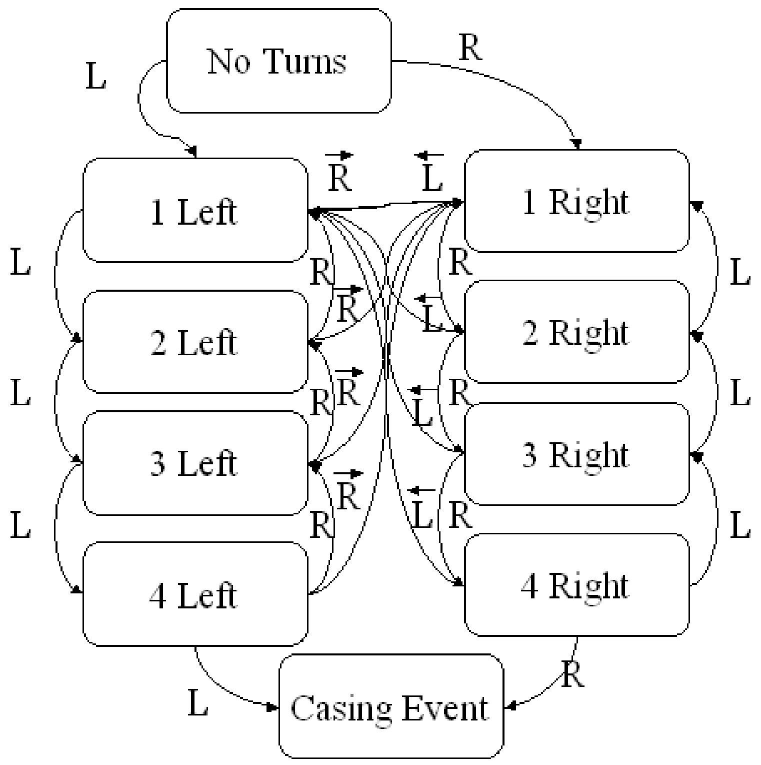


Figure 3.5: The finite automata used to detect casing events

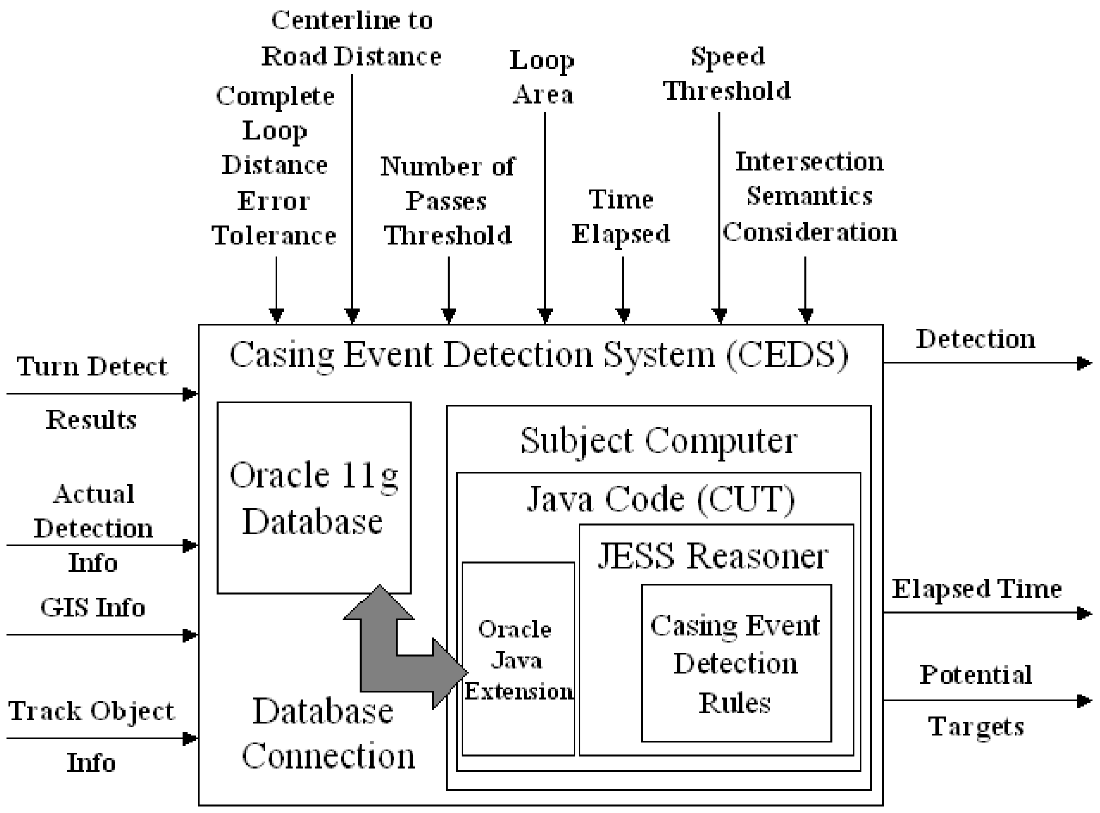


Figure 3.6: A block diagram of the Casing Event Detection System (CEDS)

Figure 3.6. Along the left side of Figure 3.6 is the workload which will be discussed in Section 3.4.4. Along the top of Figure 3.6 are the parameters for the CEDS which will be discussed in Section 3.4.5. System components include the subject computer which holds Java code for the CEDS algorithm. Within the Java code are extensions that allow SQL calls to a separate Oracle database, and the Jess reasoner along with the code depicting the rules to model the casing event. Along the right side of Figure 3.6 are the outputs of the CEDS. These include whether a detection has occurred or not, the possible buildings involved in the surveillance, and how much time elapsed during the casing event.

3.4.3 System Services. CEDS's purpose is to detect casing events. It is assumed the CEDS will not malfunction, and thus return a result of detection or non-detection as compared to the truth value. The performance of the system will be measured on the probability of detection (P_D) as calculated by Equation A.3 and the probability of a true detection (probability a detection is real or P(Real Event—Detected)) by Equation A.5. The probability a detection *is real* is similar to the commonly used probability of false alarm (P_{FA}) and is $1 - P_{FA}$.

$$Probability\ of\ Detection(P_D) = \frac{Number\ of\ positives}{total\ number\ of\ actual\ true\ events} \tag{3.1}$$

$$P(RealEvent|Detected) = \frac{Number\ of\ positives}{Number\ of\ positives + Number\ of\ false\ positives} \tag{3.2}$$

3.4.4 Workload. The workload consists of the GPS data tracks, combined with the results of the point related semantics to include turn detection. This will be combined with a truth of which casing events actually exist in the data. Specifically, the CEDS will use 8 of the 14 GPS vehicle tracks available which constitutes 17 hours of total drive time by 8 different drivers. Only 8 vehicle tracks were used because of the time consuming process that exists in manually establishing the true casing events within the 25.5 hours of track time.

3.4.5 System Parameters. Table 3.1 lists the parameters of the CEDS. For each track detection, the system will decide whether or not a casing event has occurred. To understand the *Complete Loop Distance Error Tolerance*, it is first important to understand the *Complete Loop Distance*. Figure 3.7 pictorially shows the *Complete Loop Distance* as the distance between where the first and fifth turns occur. Because we want our CEDS to detect casing events that repeat the same route, not just make turns in the same direction, we need to check the *Complete Loop Distance* when a series of turns is found. If the *Complete Loop Distance* is not sufficiently small, as defined by the *Complete Loop Distance Error Tolerance*, then it will not be considered a casing event. If a *Complete Loop Distance Error Tolerance* is set too low, then the CEDS might miss legitimate casing events. Consequently,

Table 3.1: Parameters that affect the performance of the CEDS

Name	**Description**
Complete Loop Distance Error Tolerance	The distance between the point of the first turn, and the point of the fifth turn, and what distance error is tolerable and still considered to be the same intersection.
Centerline to Road Distance	The distance between a road centerline and a point. If the distance between the centerline and a point is less than the Centerline to Road Distance, the point will be considered as being on the road.
Number of Passes Threshold	The number of times a track passes by the point of interest.
Loop Area	The area of the polygon that is formed by the consecutive turns.
Time Elapsed	The elapsed time from the beginning to the end of the casing event.
Speed Threshold	The speed at which the vehicle travels during the casing event.
Intersection Semantics Consideration	Whether or not a turn being done in an intersection is important or not. If it is, only turns that happen at intersections will transition the state of the machine, otherwise, all turns will transition the machine state whether they occur in an intersection or not.

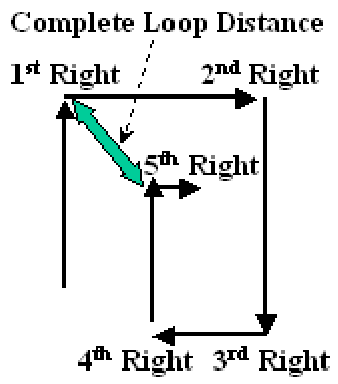


Figure 3.7: An example of five right turns used to help define *Complete Loop Distance*

if a *Complete Loop Distance Error Tolerance* is set too high, then it is more likely for false casing events to be detected. The key is to find the right tolerance for the workload that is being used. In the case of an urban environment with vehicle tracks, we expect the *Complete Loop Distance Error Tolerance* to be about the distance from one corner of an intersection to the other.

The only other parameter that needs additional explanation other than what is listed in Table 3.1 is the *Intersection Semantics Consideration* parameter. This parameter is a toggle to tell the CEDS whether we want to consider all turns, or only those turns that are in intersections as valid turns that will transition the state of the finite automata depicted in Figure 3.5. So when *Intersection Semantics Consideration* is set to "YES," only turns that occur in intersections will transition the state of the finite automata. When *Intersection Semantics Consideration* is set to "NO," all turns regardless of whether they occur in intersections or not will transition the state of the finite automata.

One other assumption is that the *Centerline to Road Distance* also defines the boundary of what determines an intersection. Thus, an intersection is defined as within the *Centerline to Road Distance* of two roads. An example of this would be if the *Centerline to Road Distance* were set to 15m, then an intersection would be

Table 3.2: Levels of the *Centerline to Road Distance* factor to be tested

Level	**Value**
1	5m
2	10m
3	15m

Table 3.3: Levels of the *Complete Loop Distance Error Tolerance* factor to be tested

Level	**Value**
1	15 meters
2	30 meters
3	60 meters

defined as a 30m by 30m square centered on the crossing of those two centerlines (30m representing 2×15m).

3.4.6 Factors. The factors that will be varied in this experiment are the *Complete Loop Distance Error Tolerance*, the *Centerline to Road Distance*, and the *Intersection Semantics Consideration*. Each factor will be varied between experiments as shown in Tables 3.2 - 3.4.

Standard roads in the United States have a lane width of 11ft (3.35m) [15]. It follows then that two lane roads are 22ft (6.7m) wide, four lane roads are 44ft (13.4m) wide, four lane roads with a turn lane are 55ft (16.76m) wide, and six lane roads with a turn lane are 77ft (23.47m) wide. As a result of these different possible road sizes, the levels of 5m, 10m, and 15m for the edge of the road to center are appropriate as listed in Table 3.2.

It follows for the size of intersections, that if roads are between 6.7m wide and 23.5m wide, that the distance from one corner to the opposite corner would be $\sqrt{2} \times width$ or 9.5m to 42m. As a result, we would expect our *Complete Loop Distance Error Tolerance* to be near those numbers. Adding an additional 50% for error gives us the values that are the top and bottom values in Table 3.3.

Table 3.4: Levels of the *Intersection Semantics Consideration* factor to be tested

Level	**Value**
1	Ignore turns not in intersections
2	Consider all turns

Table 3.5: Configuration of the CEDS under test

Computer Specifications	Intel Xeon 3.2 GHz, 3 GB RAM
Operating System	Windows XP 64-bit Service Pack 2
Java JDK Version	6
Jess Version	7.1p1 8/6/08
Oracle Version	Oracle Spatial 11g
JDBC Driver Version	5
OSU data set collection dates	28 Apr 2006, 28 Oct 2007

With regards to the *Intersection Semantics Consideration* factor, it could be argued that casing events that only consider turns in intersections are more accurate, while others might suggest turning into parking lots and looping around buildings in this manner are potentially more suspicious. As a result, we tested both configurations.

3.4.7 Evaluation Technique. The purpose of the experiment is to determine the accuracy of the CEDS. Direct measurement was the technique used because data for direct measurement is available in the form of GPS tracks, and direct measurement is highly favored because of the increased believability of the results. Table 3.5 lists the configuration of components in the CEDS.

3.4.8 Experimental Design. A full factorial will be run using the levels discussed in Section 3.4.6 resulting in 18 different configurations tested.

3.5 Casing Event Detection Methodology Summary

Extending the results from Turn Detection in Section 3.2.3 using the best turn detection configuration and using the semantics developed in Section 3.3 we extended the events that the system will detect to include casing events. This experiment tests whether intersection semantics are important in detecting casing events, what distance is best to use for defining that intersection, and what loop distance to use when detecting casing events. It will also test the finite automata shown in Figure 3.5 for its accuracy in detecting casing events. The goal is to find the levels that result in the highest probability of detecting a casing event with the minimum probability of having a false positive result.

IV. Casing Detection Results and Analysis

In this chapter we will present the results of the Casing Event Detection experiments, and arrive at a combination of factors that best detects a casing event. We'll further extend those results to provide more information to the user on detected events to include how much time was spent in the casing event, the address of the likely target, and the number of times a vehicle has passed by the target.

4.1 Results and Experiment Discussion

The methodology from Section 3.4 was applied to produce the results for the Casing Event Detection System (CEDS). Results from initial experiments led to a series of experiments that further refined the best performance of the CEDS. These series of experiments will be discussed in subsequent sections.

4.1.1 Preliminary Results.

The first experiment demonstrated that the approach using the finite automata in Figure 3.5 missed several casing events. When run on a subset of the data only 106 minutes long, the algorithm detected only 1 of 10 casing events, and had several false positive results as well. A new approach had to be developed.

4.1.2 Revised Approach.

A revised approach was developed in order to improve the performance of the system. Instead of detecting a casing event from five turns, an approach using three turns was implemented to improve the detection capabilities of the system.

4.1.2.1 Revised Finite Automata Model for Case Event.

The Jess reasoner is used to build a finite automata (FA) as shown in Figure 4.1 to detect a casing event. As with the FA before, each input of a Left turn (L) or right turn (R) will transition from one state to another state. Then, when a series of three turns in the same direction is detected, the code begins tracking forward from the point of turn three, and backward from the point of turn one, in order to find a

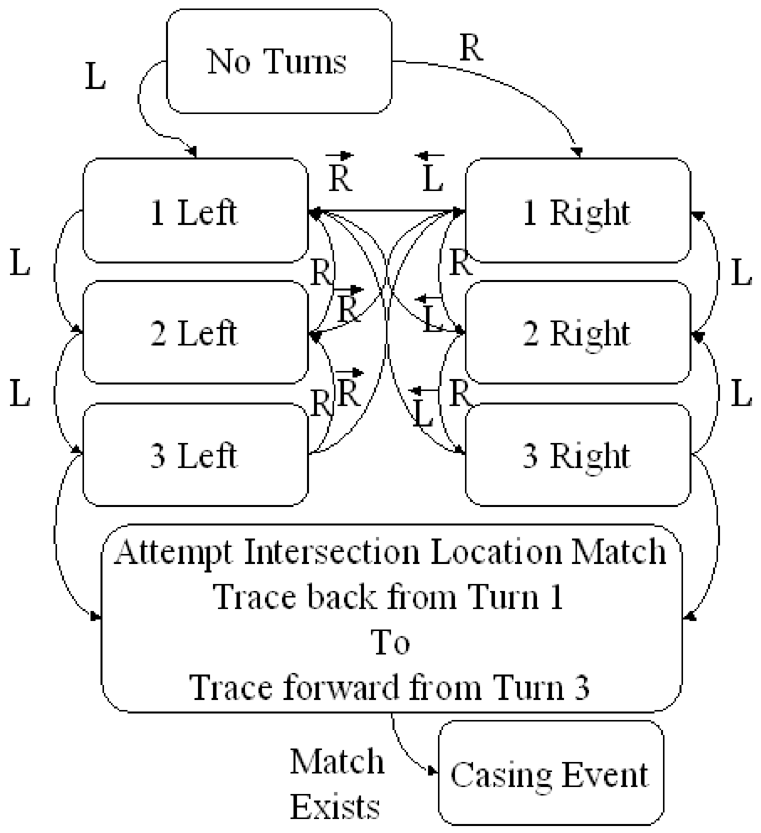


Figure 4.1: A revised finite automata used to detect casing events

location in common under the *Complete Loop Distance Error Tolerance*. To improve performance, the algorithm only looks a set time period in the past and the future in order to find a portion where the tracks cross. In addition, only points that are in intersections are checked further improving performance. The concept is shown pictorially in Figure 4.2. The time period that represents the amount of time to look forward and backward became a factor in the system called *Time Match Threshold* and was tested at the levels shown in Table 4.1.

Table 4.1: Levels of the *Time Match Threshold* factor to be tested

Level	**Value**
1	5 minutes
2	3.75 minutes
3	2.5 minutes

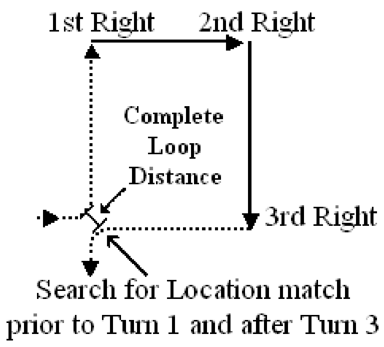


Figure 4.2: A three turn approach to detecting casing events. Searching for a location match traces forward from the 3rd turn and backward from the 1st turn comparing all points in intersections until one is found less than the *Complete Loop Distance Error Tolerance*. The *Complete Loop Distance* is depicted as the distance between the past and future tracks.

4.1.3 Experiment #1: Determining Loop Distance and Intersection Semantics.

The first experiment tested and varied the factors (as shown in Tables 3.2, 3.3, 3.4, and 4.1) over a 106 minutes tracklet. Prior to experimentation, this data set had 6 casing events identified in it from a human visual inspection. After experimentation, it was discovered that there were actually 10 casing events in the data track. The first experiment was run setting *Time Match Threshold* to 5 minutes. Referring to the results in Table 4.2, it was discovered early on (compare Line 1 and Line 2) that restricting those turns to only turns at intersections missed half of the casing events. The root cause of this was that some casing events occurred in areas of the map that were not roads, but were alleyways instead. Since the alleyways didn't have names, they were not tagged as roads, and as a result where they met with roads were not considered intersections. It was further discovered (compare Lines 2, 3 and 4 in Figure 4.2) that reducing the *Loop Distance* to 15 meters reduced the number of detections, and increasing it to 60m had no effect. In addition, it was discovered that varying the *Intersection Semantics Consideration*, and the *Loop Distance* did not have any effect on execution time. These tests showed that further tests would likely have the best results if the *Loop Distance* were set to 30m, and the *Intersection*

Table 4.2: Results of Casing Detection varying *Loop Distance* and *Intersection Semantics*

Line No.	**Inter-section Dist**	**Loop Dist**	**Inter-section On**	**Time match** (min)	P_D (%)	**Prob Real** (%)	**Exec Time**
1	15m	30m	Y	5	50	50	7 min
2	15m	30m	N	5	100	83.3	7 min
3	15m	15m	N	5	90	81.8	7 min
4	15m	60m	N	5	100	83.3	7 min

Semantics Consideration were turned off, considering all turns whether they were in intersections or not.

4.1.4 Experiment #2: Varying Time Match Threshold and Improving Performance Time. The second experiment tested and varied the *Time Matching Threshold* factor over a 106 minute tracklet. Table 4.3 shows the results of varying the *Time Matching Threshold*. The results show that decreasing the *Time Matching Threshold* decreases the execution time. If decreased too much, it can have a negative effect on the detection probability. Further testing was accomplished between 3.75 min and 2.5 min *Time Matching*, to determine the behavior between those values. The results show the same trend that decreasing the *Time Matching Threshold* decreased execution time, but probability of detection was tightly related to the specific data set that was being tested, so the exact figure is only beneficial in speeding up performance on a specific data set. Since the focus is on trying to determine parameters that work for many different locations and driving patterns, not just one driver in one location, the pursuit of minimizing execution time by lowering *Time Matching* less than 3.75 minutes was abandoned. Further research can be applied in this area if the benefit of this performance gain is essential, and the locations and driving patterns are well constrained.

4.1.5 Experiment #3: Varying Intersection Distance and Improving Performance Time. The third experiment tested and varied the *Centerline to road*

Table 4.3: Results of Casing Detection varying *Time Matching*

Line No.	**Inter-section Dist**	**Loop Dist**	**Inter-section On**	**Time match** (min)	$\mathbf{P_D}$ (%)	**Prob Real** (%)	**Exec Time**
1	15m	30m	N	5	100	83.3	7 min
2	15m	30m	N	3.75	100	83.3	4.25 min
3	15m	30m	N	2.5	60	75	2.15 min

Distance factor over a 106 minute tracklet. Table 4.4 shows the results of varying the *Centerline to Road distance* (and likewise the *Intersection Distance*). The results show that decreasing the *Intersection Distance* decreases the execution time drastically at the expense of decreasing the probability of detection.

Table 4.4: Results of Casing Detection varying *Intersection Distance*

Line No.	**Inter-section Dist**	**Loop Dist**	**Inter-section On**	**Time match** (min)	$\mathbf{P_D}$ (%)	**Prob Real** (%)	**Exec Time**
1	15m	30m	N	3.75	100	83.3	255 sec
2	10m	30m	N	3.75	90	81.8	75 sec
3	5m	30m	N	3.75	50	83.3	13 sec

The execution time of the casing detection algorithm is driven by the number of distances that need to be calculated in matching the past with the future. This is why decreasing the *Time Matching* and decreasing the *Intersection Distance* decreases execution time, because less points to compare mean less distance calculations. Since there are multiple points within an intersection, and we really only need to check one point in the past with one point in the future for each intersection in the past and future, it was thought that it were possible to reduce the number of computations required by reducing the number of points that are within an intersection. While more sophisticated methods might be possible, the approach that was taken was tagging all points within 15m of an intersection, and then not using those points less than a certain distance. In this way, we compare points only entering or leaving and intersection, but not all the points in the middle of one. Table 4.5 shows those results. It shows that comparing only those points that fall between 15m and 10m yields a four-fold performance gain without any loss of accuracy. Further testing comparing those points that fall between 15m and 13m yields a nine-fold performance improvement without much loss in probability of detection. Applying these results

Table 4.5: Results of Casing Detection varying *Intersection Distance*

Line No.	**Inter-section Dist**	**Loop Dist**	**Inter-section On**	**Time match** (min)	$\mathbf{P_D}$ (%)	**Prob Real** (%)	**Exec Time**
1	15m	30m	N	3.75	100	83.3	255 sec
2	15m-10m	30m	N	3.75	100	83.3	66 sec
3	15m-13m	30m	N	3.75	90	81.8	15 sec

to a real world situation, this means that we could track about 96 vehicles with high accuracy, and 424 vehicles with slightly diminished accuracy in real-time. Further analysis was done to determine the speed that a vehicle would have to travel in order to have a detection missed in the 15m to 10m zone on each side of an intersection. At a detection rate of one detection every 0.5 sec, the resulting speed is 22 miles per hour. This means that there is a possibility if a vehicle is traveling over 22 miles per hour that a detection will not occur in the 15m to 10m zone. Since it is very likely that vehicles are traveling over 22 miles per hour through an intersection, we tested these parameters over a larger data set to evaluate if any detections are being missed because of the *Intersection Distance* value of 15m less 10m.

4.1.6 Experiment #4: Testing against more data. The fourth experiment tested whether the levels for the factors determined for a subset of the data applied to a larger data set. The first data set that was used for Experiments #1 - #3 used a single track of 106 minutes and 10 casing events. In this experiment the data set included eight different tracks of 17 hours and 43 casing events. Table 4.6 shows the configuration of the experiment and the results. All 43 casing events were detected and only four false positives were found.

Table 4.6: Results of Casing Detection on 17 hours and 8 tracks

Line No.	**Inter-section Dist**	**Loop Dist**	**Inter-section On**	**Time match** (min)	$\mathbf{P_D}$ (%)	**Prob Real** (%)	**Exec Time**
1	15m-10m	30m	N	3.75	100	91.5	10.5 min

Table 4.7: Number of casing event detections by vehicle track

Track Name	**Casing Events Detected**
AP_A	10
L1A	1
L1B	12
L2A	6
L2B	0
L3A	11
OM	3
DB	0

Table 4.7 lists how many casing events were detected on each track. From this we can also get an idea on which vehicle tracks are likely to be suspicious. Tracks *AP_A*, *L1B*, and *L3A* are likely to be doing suspicious activity as they each have over 10 casing events. Other vehicle tracks like *L2A*, *L1A*, and *OM* have less casing events, so more information may be needed to confirm them as true suspicious vehicles or not.

4.1.7 Analysis. A binomial analysis was run on the results shown in Table 4.6 to determine a 95% confidence interval for detection of a casing event. Since all events were detected a confidence interval cannot be found for the probability of detection. The results of the analysis of the probability that a detection is real show that while we estimate the probability of detection at 91.5%, we can, with 95% confidence, say that it is between 83.54% and 99.46%. This means that given another data set, we can expect 95% of the time that the average probability a detection is real will be between 83.54% and 99.46% on that new data set.

Even though we are only using points that are between 15m and 10m of an intersection, and there are several instances where vehicles are driving through intersections at greater than 22 miles per hour, the number of missed casing events did not increase. As a result, trimming down to only those points within 15m and 10m of an intersection is not as big of concern as once thought.

4.1.8 Casing Event Detection Results Summary. In this section we have demonstrated the ability to write and test an algorithm that detects casing events for GPS vehicle tracks. By using a reasoner to detect a series of turns that have more than three turns in the same direction ignoring whether those turns are in an intersection, then looking back 3.75 min from turn one and comparing those detections within 15m and 10m of an intersection with those points 3.75 min after turn three and within 15m and 10m of an intersection and returning results that are less that 30m away from each other. In our testing we detected 100% of casing events, and can say, with 95% confidence, that the probability that a detection is real falls between 83.54% and 99.46%.

4.2 Extending Casing Events Detection

Now that we have a detection system that detects the outward behavior with casing it is possible to move further up into the knowledge layer (Figure 3.2) to add specificity to the casing event (what building or address is being cased) and credibility to the detection (number of times the vehicle has passed this location, and how much time elapsed). Section 4.2.1 will discuss calculating the time duration of a casing event and its value. Section 4.2.2 discusses a few possible approaches to deriving a target building from the casing event detection, and implements one such approach. Section 4.2.4 discusses how to derive a street address given a target building. Finally, Section 4.2.3 will discuss counting the number of times a vehicle has passed a particular target building and its applicability to determining whether casing events are significant or not.

4.2.1 Time Duration.

In the output of the casing detection, it is known what waypoints are included in the casing detection. By comparing the start point to the end point we are able to calculate the duration of the event. This value may be useful in determining whether a casing detection is a true surveillance activity or not. In general, actors conducting surveillance watch targets for a considerable amount of time [53], so a casing event that lasts only a few minutes is not likely a true surveillance activity. We can also use this time duration to help determine what building might be targeted along the route.

4.2.2 Target Building.

There are two major approaches to determining a target building given the casing detection. One approach is to define a geometry using the casing track (like a polygon with the roads traveled as the boundaries of the polygon) and determining what buildings interact with that geometry. Another approach is to retrace the track and determine what buildings are on that track on either side of the road. There are benefits and drawbacks to each approach. These approaches help us in narrowing down a group of possible buildings. Further computation needs to be done to determine a rank order of those buildings to help determine the most probable targets.

4.2.2.1 Geometry Approach.

There are several ways that geometries can be derived from the casing detection. One method is to draw a simple rectangle using the opposite corner points of a track. Another method is to use a convex hull approach to include the boundary of the entire track. Of additional concern is how best to apply the geometry approach and consider buildings that are along the track, but outside the shape that is formed in the boundary of the casing route.

The rectangle approach is a very simple and computationally fast approach when comparing to other layers (i.e. buildings). Its drawbacks are that it likely ignores buildings that are on the outside of the casing route, and may miss buildings if the dimensions of the casing route are not perfectly square, or lined up square with coordinate lines (like latitude and longitude).

The convex hull approach takes more time to compute than the rectangle approach, both in calculating the convex hull, and in comparing the resulting non-square geometry to other layers. It does fix the problem that the rectangle approach has with casing routes that are not square and not lined up squarely with coordinate lines. It still has the shortfall that without additional extension, it only considers buildings within the casing route, and ignores buildings that are on the route, but on the outside of the route.

To fix the problem that both geometry approaches have with buildings that are outside the route, an arbitrary distance value could be used to extend the geometries to include those items that are a certain distance from the geometry defined by the casing route. This fixes the problem of ignoring buildings outside the route, but is highly dependent on the arbitrary distance value chosen. We will discuss an approach to picking an appropriate value in detail in Section 4.2.2.2.

There is another concern of using a geometry approach. For large area casing events, the resulting geometry is very large, forcing all buildings within that area to be considered. For a large area, it is more likely that the buildings near the edges are the buildings being surveilled than it is for a building in the middle of the geometry. This means that for large area casing events, more buildings are considered than potentially should be.

4.2.2.2 Retrace Approach. A retrace approach retraces the casing route and considers all buildings along that route. There are two ways to define a building being on that route. One definition is what buildings can be seen from each point along the route which we will call a line-of-sight retrace approach. Another definition is what buildings are within a certain distance of the road being traveled on regardless of whether they can be seen or not. In general, retrace approaches are slower computationally than geometry approaches because of the number of comparisons that need to be made. Each retrace approach has its drawbacks and benefits that we will discuss in more detail.

Using a line-of-sight approach is the most computationally expensive approach, but arguably the most accurate approach in terms of our application. Using this approach would mean that we would only consider those buildings that can actually be seen from the casing route. The driver behind the expensive computation is the number of comparisons that need to be made in 3-dimensional (3-D) space. For each waypoint a 3-D vector is produced between the waypoint and each of the eight corners of every building in the area. Then each vector is compared to each other beginning with the shortest vectors (nearest buildings) with the goal of determining which vectors need to be removed from the overall set of vectors. When all vectors have been compared and either kept or removed, the remaining set of vectors defines which buildings are in the line of sight of the waypoint. Considering our example of 25.5 hours of driving time and 140,000 waypoints and an estimated 1000 buildings this means that 8.96 trillion ((8 vectors $\times$ 1000 buildings)^2 $\times$ 140,000 waypoints) comparisons would need to be made. Since this is very computational for a simple problem it makes it not an attractive choice for determining a target building.

Using a retrace approach that identifies buildings with in a certain distance from the route is a faster approach than calculating line-of-sight. The key element in making this kind of retrace approach work effectively is finding the appropriate distance to use for the area. Because the appropriate distance should be related to the area that we are in, it is necessary to derive it from existing data related to the area. The approach that we used was to calculate the distance from each building to its nearest street. We then aggregate that to determine the average and standard deviation distance for a building to its nearest road. For the data set in the OSU area, the average distance was 38.5m with a standard deviation of 38.1m.

4.2.2.3 Determining target building through max time interaction.

Once we defined a subset of buildings that are involved in the casing event, we can further determine what buildings within that subset are more of a focus of the surveillance. Since the purpose of surveillance involves the amount of time being spent in view of a target, we will use an approach that calculates the amount of time that was spent during the casing route on each building. We then can divide the amount of time spent in front of each building by the total elapsed time to determine what percentage of time was spent in front of a particular building. We can then use this to rank order buildings based on the amount of time that was spent in front of them. If further screening is required on casing events, we can use both the percentage of time, and the total amount of time in front of a building as discriminators in helping us determine actual surveillance activities from the casing events that are detected by the system.

There are a few concerns with using this type of approach. One concern is that buildings that are larger are more likely to appear that they have a larger amount of time spent near them. Also buildings that are closer to the road are going to appear that more time is spent near them. Both these cases occur because we total the amount of time based on a certain distance from the building. As a result, larger buildings take up more geographical space, and are more likely to have more detections near them. This is also the case with buildings that are closer to roads where they have more road-space within their distance threshold than buildings that are farther from roads, making buildings that are closer to roads appear that more time is spent in front of them.

In an urban area such as OSU that has several buildings, it is best for us to use as small a distance as possible. While this runs the risk of missing the actual building which is being watched because it is off the road a ways, detections will occur for buildings that are closer and still yield the correct detection address which can be used to find the building that is off the road. This is preferred over using a large distance which will drive results to be centered in the casing area, which may not be the true target. It is also more difficult to resolve a correct address from the center of a city block, than it is from the edge. This is because if a point to be resolved is in the middle of a city block, it can map to four possible different streets, where if it at

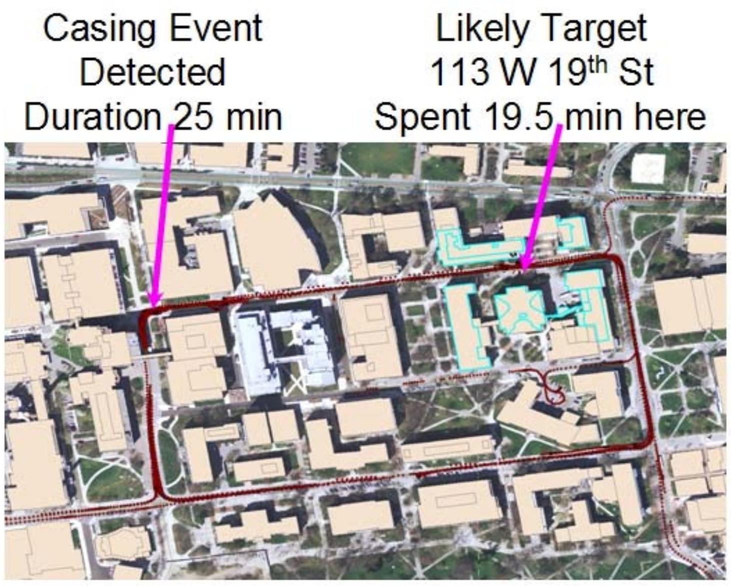

Casing Event Detected Duration 25 min
Likely Target 113 W 19th St Spent 19.5 min here

Figure 4.3: A screen shot of casing detection and determining which building is the target of surveillance

an edge, it maps to one street. While the street number may be off a little, this still gives a better translation than an incorrect street.

Testing was done on the OSU data for casing event detection using 38m (average), 76m (avg + stdev), and 114m (avg + 2 StDev). Evidenced by the increase in the number of predicted locations within a circled route, and those locations near corners, testing confirmed the concerns about larger values skewing detections to the center of a casing route and the corners of the casing routes. Testing also showed that having a small distance does not misinterpret any results or miss the intended target buildings. For this reason we used 38m (the average road to building distance for the area) in determining which building is the most likely target. Figure 4.3 shows some results of the casing detector then concluding what group of buildings is most likely the source of the surveillance.

4.2.3 Number of Passes. An additional discriminator that could be used in determining whether a casing event is indicative of a true surveillance activity is the number of times a vehicle passes by a building of interest. If we have decided that a building is being watched, then we can count the number of times that a vehicle has passed by that building and at what times. This information could be aggregated over time just as the amount of time spent in front of a building could be aggregated over time to give a more complete view of how much time a target is spending in different locations, and what buildings are related to this suspicious vehicle. This information could be used by a higher level process in determining whether or not a vehicle is acting suspicious or just has casing behavior but is not actually suspicious. An example of this might be someone that is lost and looking for an particular location circling around until they find it, as opposed to a suspicious act. The lost individual may circle only four times, while a suspicious individual my pass a location over ten times.

4.2.4 Address Lookup. Once a building of interest is identified, it would also be helpful in determining the address of the building to be able to find it on a map, or relate it to other objects by address. First we calculate the centroid of the building's geometry and get its latitude and longitude coordinates. Next we can use a reverse geocoding lookup web site [25] to translate from latitude/longitude into a street address.

4.2.5 Use of Extended Casing Results at Higher Knowledge Levels. The results of the CEDS and the extension of those results including the duration of the casing event, the target building with its address, and the number of passes is information that can become inputs to higher level systems focused on determining which casing events (as we have defined them) are true surveillance activities. For example, it may be possible to take several series of CEDS results each with time duration, address, and number of passes and build a learning algorithm that determines what time duration and number of passes for particular addresses are significant and which ones are not. This could form a baseline of casing behavior to be used to determine the finer threshold between true surveillance activities and innocent surveillance-like behavior.

4.3 Summary of Casing Event Detection

The results presented in this chapter lead us to the following conclusions regarding casing event detection. First, we found out that the first approach (Figure 3.5) using five turns did not work, but an approach with three turns and route-trace-matching did (Figure 4.1 and Figure 4.2). Next, we discovered that the use of semantics in only considering turns that were in intersections was not helpful and all turns should be considered whether they occur at intersections or not. We verified that 30m was an appropriate distance for matching up loops considering the size of intersections discussed in Section 3.4.6 for an urban area. Further, we confirmed a route-trace approach matching only those points that were in intersections because they kept high accuracy in detection and decreased execution time. Also, we discovered that tracing back longer than 2.5 min was necessary, and tracing for 3.75 min was sufficient. Lastly, we discovered that decreasing the intersection distance from the center of the intersection, while it saved time, it decreased accuracy. A better approach that only compared points within 15m and 10m saved execution time and maintained 100% probability of detection with a 91.5% probability that the detection was real (95% confidence interval from 83.54% to 99.46%). In this 100% detection mode, it is estimated that 96 vehicles could be tracked simultaneously in real time. Using an intersection distance between 15m and 13m maintained a 90% probability of detection and 81.8% probability that a detection was real and sped up execution time to the point that about 424 vehicles could be tracked simultaneously in real time.

Once a casing event was detected, additional analysis can be done to provide more information to the user of the system. This would include a description of the time involved, a likely target based on where the majority of time was spent, the number of times that vehicle has passed the location, and the address of the location of interest. These outputs can be used to help filter out those false alarms of an individual driving loops around the block, that are not actually casing a location.

V. Automated Social Network Construction Methodology

Social networks are a beneficial analysis tool in surveillance and counter-surveillance activities as was discussed in Section 2.5.2. The shortfall of applying social networks to counterterrorism is that social networks often take considerable time and resources to construct. An automated system that can process persistent video from a 24-hour overhead source, track vehicles to generate vehicle tracks, and relate the vehicle tracks to each other would construct a social network to free analysts for other critical tasks. In addition, such a system could further allow for the constructing of social networks on individuals before they are identified as suspicious, saving the lead time required to construct them. In this way, if a suspicious individual is identified, an intelligence analyst can proceed immediately with checking who the individual has interacted with and where they have been operating based on the persistent video and the social network constructed by the system.

5.1 Problem Definition

The problem addressed is applying information contained in the GPS vehicle tracks, along with other GIS information (buildings, roads) to construct a visualization of a social network that represents the interactions tracks had with other tracks and with buildings. For example, two vehicles rendezvous in a parking lot for several hours and one of them later robs a bank (see example rendezvous in Figure 5.1). The parking lot interaction would clearly be information valuable to an investigation on the bank robbery.

The approach was to use parameters as defined in Section 5.3 to define interactions between vehicles and buildings. These interactions form the basis for populating links from vehicles to vehicles and vehicles to buildings in a social network. This network is then displayed and evaluated for its value to understanding the operations that occurred.

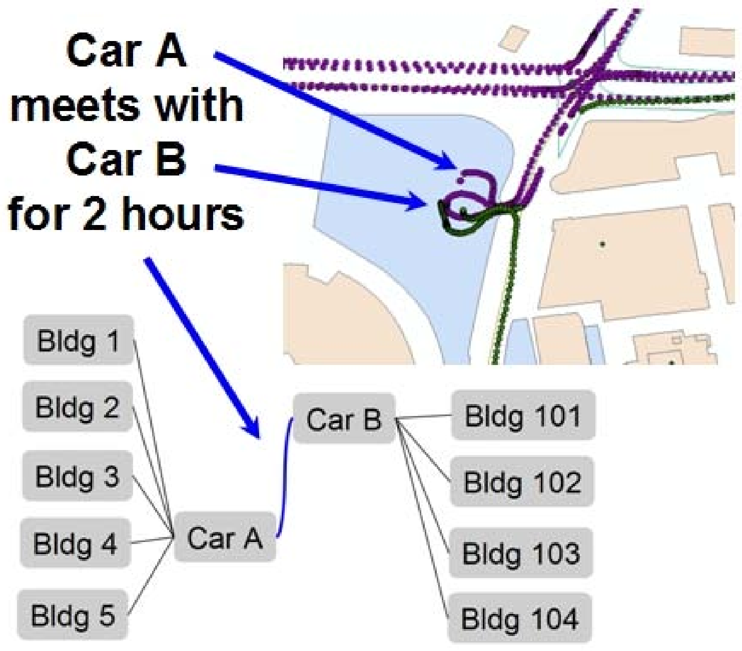


Figure 5.1: An example of a vehicle-to-vehicle interaction and resulting graph representing a social network

5.2 Workload

The workload consists of the GPS data tracks, combined with the results of the point related semantics to include turn detection as discussed in Chapters III and IV. The same discussion os Section 3.2.2 applies as to why GPS tracks are an appropriate workload representing persistent video tracks. More information will be required than has been used in previous tests as more relationships need to be built. As a result, we used all 14 GPS vehicle tracks available which constitutes 25.5 hours of total drive time by 14 different drivers.

In addition, it is known that of the 14 different tracks generated, some vehicles worked together in completing their assigned scenarios, while other vehicles did not interact. It is not known ahead of time what scenarios were accomplished, only that some were, and some tracks were designated as “random driving.” This is mentioned because the workload does include both elements of known interaction, and elements of known non-interaction that should be verifiable after the social network is constructed.

5.3 Parameters and Definitions

In order to build interactions between vehicles and between a vehicle and a building, we must first define the parameters that will be used in time and space to describe these interactions. Links between vehicles will be built when two vehicles have stopped in the same location for a set period of time. Likewise links between a vehicle and a building will be built when a vehicle has stopped near a building for a set period of time. The following sections define the necessary length of time and distance for these links.

5.3.1 Definition of a stop. Defining a stop is the first necessary step. A stop will be defined as when a track stays within a 6m radius for over 30 sec. The 6m distance allows some error in tracking, but not more error than the approximate length of a car. Thirty seconds is likely too accurate as it may also include long stoplights, but data can be restricted further when individual queries are made. These parameters were generally tested with the data that was used and may need to be verified or tweaked in order to apply to the specifics of an area.

5.3.2 Defining vehicle-to-building links. Vehicle-to-building links will be constructed whenever a vehicle stops in front of a building. There are two approaches to determine which building is being stopped in front of: a Nearest Neighbor approach, and a distance approach.

5.3.2.1 Distance Approach. A distance approach would take a vehicle track, and find all buildings within a certain distance of where that vehicle track stopped. While this is computationally more efficient than the Nearest Neighbor approach, it is very dependent on a consistent distance between stop points and buildings. A distance threshold that is set two low returns no vehicle-to-building links. A distance threshold that is set to high returns several buildings per stop, increasing the clutter on the social network.

5.3.2.2 Nearest Neighbor Approach. A Nearest Neighbor approach would take a stopped vehicle track, and find the building that is closest to that point. This approach is the most straight forward and best approach as it is flexible to different operating environments (those that have lots of buildings, and those that have few), and reduces each stop to have only one building which is helpful in keeping the social network uncluttered with unnecessary and redundant information as might occur with a distance approach. Those areas that have very great distances between buildings (i.e. rural areas, or 3rd world countries), a Nearest Neighbor approach may pick out a building that is nowhere near the actual stop. In these cases it might be best to take a different approach all together than defining vehicle-to-building links. One suggestion would be sectioning areas off in a grid and populating "vehicle-to-area links" instead of vehicle-to-building links. In the cases where an area is variable (some with buildings in pockets, and some areas without buildings) it would be best to aggregate buildings and areas together. Behavior could be modeled more exactly to buildings when they are there, and more generally to areas when there are no buildings.

In constructing vehicle-to-building links, we will use a Nearest Neighbor approach for stops that are longer than five minutes. This choice of duration is arbitrary and can be set higher or lower depending on the granularity of results that are required. The choice of using a Nearest Neighbor approach is deliberate to include at least one building per stop, and not over clutter a network with several buildings per stop.

5.3.3 Defining vehicle-to-vehicle links. Vehicle-to-vehicle links will be constructed when two vehicles stop for longer than five minutes within 100m of each other. The choice of time again is arbitrary and can be set higher or lower depending on the granularity of results that are required. Setting the distance to 100m is used as an estimate of a typical parking lot size. When data sets get large, this distance will likely have to be lowered to decrease computation required and increase the speed of

Table 5.1: Time and Distance Definitions for a stop, vehicle-to-vehicle link, and vehicle-to-building link

Parameter	**Distance**	**Time**
Stop	< 6 meters	> 30 sec
Vehicle-to-Vehicle Link	< 100 meters	> 5 min
Vehicle-to-Building Link	nearest building	> 5 min

computing the results. Independent of the computation required, the distance should be set appropriate to the level of interaction that it is expected to be detected or modeled as an interaction between vehicles.

5.4 Variables

In Section 5.3 we defined the distances and times for stops, vehicle-to-vehicle links and vehicle-to-building links. It is important to note that the definitions that we have set for these terms are flexible depending on the location that is being analyzed. We suggest that these values are reasonable for an urban area, but in this section we will discuss how these variables will likely perform in different locations. Table 5.1 summarizes these definitions and we will use this as a basis to discuss how the system is likely to perform in urban areas as well as rural areas. We can further define each parameter and its associated distance and time thresholds as variables listed in Table 5.2.

Table 5.3 describes how increasing or decreasing the variables would likely affect the results of the visualization of the social network. In addition to the results listed in Table 5.3 its important to note the effects of using the Nearest Neighbor approach to vehicle-to-building Links in an urban environment. The Nearest Neighbor approach will work best when there are many possible buildings. When there are few buildings, then links may be populated when a vehicle stops several hundred yards or even as far as miles (if that is the nearest building) making the link produce an interaction which is probably not representative of true behavior.

Table 5.2: Definition of Social Network Variables

Variable Name	**Description**
Stop Distance Threshold	The radius distance that a series of waypoints forming a track must stay within in order to be considered in the same place.
Stop Time Threshold	The duration of time that a series of waypoints forming a track must stay within the Stop Distance Threshold radius in order to be considered a stop.
Vehicle-to-Vehicle Distance Threshold	The distance that two stops must be under in order for a vehicle-to-vehicle link to be constructed.
Vehicle-to-Vehicle Time Threshold	The duration of time that two stops must be under the Vehicle-to-Vehicle Distance Threshold and stopped, in order for a vehicle-to-vehicle link to be constructed.
Vehicle-to-Building Distance Threshold (Distance Approach)	The distance a vehicle must be under when it stops in order for a vehicle-to-building link to be constructed.
Vehicle-to-Building Time Threshold	The duration of time a vehicle must stop under the Vehicle-to-Building Distance Threshold in front of a building in order for a vehicle-to-building link to be constructed.

5.5 Social Network Construction Methodology Summary

By defining stops for a series of vehicle tracks, and being able to relate them in space and time, we hope to be able to build a social network that represents the interactions of vehicles. In this manner we can better understand groups that are working together and aid an intelligence analyst with a full network of interactions. The experiments will test whether the parameters chosen are appropriate for an urban area, and show if the approach to building a social network from vehicle tracks and GIS information is viable.

Table 5.3: Behavior of Social Network variables in various environments

Variable Name	**Result if Decreased**	**Result if Increased**
Stop Distance Threshold	Less error tolerant. More precise determining stop points. Smaller is better for urban.	May consider very slow moving vehicles as stops. If taken extreme, will not be accurate in determining precise location of stops more than a general area.
Stop Time Threshold	Pick up stops at stop signs, stops in traffic, stops at red lights, short unloading stops.	Focus on longer stops. If too large, may miss events. Larger is better for both rural and urban, but not so large that the overall event can be missed.
Vehicle-to-Vehicle Distance Threshold	Assumes small area of interaction, vehicles must be near each other for interaction to occur. If set too small, may miss interaction because of parking availability constraints. Smaller necessary for urban because many false positives of parking in same parking lot.	May get more false interactions of vehicles parked together, but not interacting. Larger is going to get more interactions in rural without the added error of false positives.
Vehicle-to-Vehicle Time Threshold	Pickup short interactions like brush-pass interactions, or quick vehicle-to-vehicle occupant changes.	Returns longer interactions like meetings.
Vehicle-to-Building Distance Threshold (Distance Approach)	Few links returned, and a possibility of missing vehicle-to-building links if the stop is in an area outside the Vehicle-to-Building Distance Threshold.	Many links returned. A very cluttered network that may lose the actual location of interaction because so many results are returned per stop.
Vehicle-to-Building Time Threshold	Picks up short interactions like drop off person at a building, or other interactions that keep the vehicle running.	Picks up longer interactions like parking.

VI. Social Network Construction Results

Following the methodology that was defined in Chapter V a series of visualizations were created representing the interactions of vehicle GPS tracks. In this chapter we will discuss how the results were implemented in Section 6.1, the resulting social network graph visualizations in Section 6.2, some enhancements to the visualizations in Section 6.3, and lastly we will analyze the meaning of results in Section 6.4.

6.1 Implementation

To display a social network, we chose Prefuse [43] as our visualization package to represent the social network graph. Prefuse was chosen because it was written in Java and came complete with several different ways to display the same graph. To use prefuse, data needed to be populated in two tables; one listing the nodes in the graph, and one representing the edges in the graph. Once these tables were populated, the full functionality of prefuse is available. We thought that it was a logical to relate both vehicles and buildings as nodes and to have edges represent the interactions between vehicles and buildings.

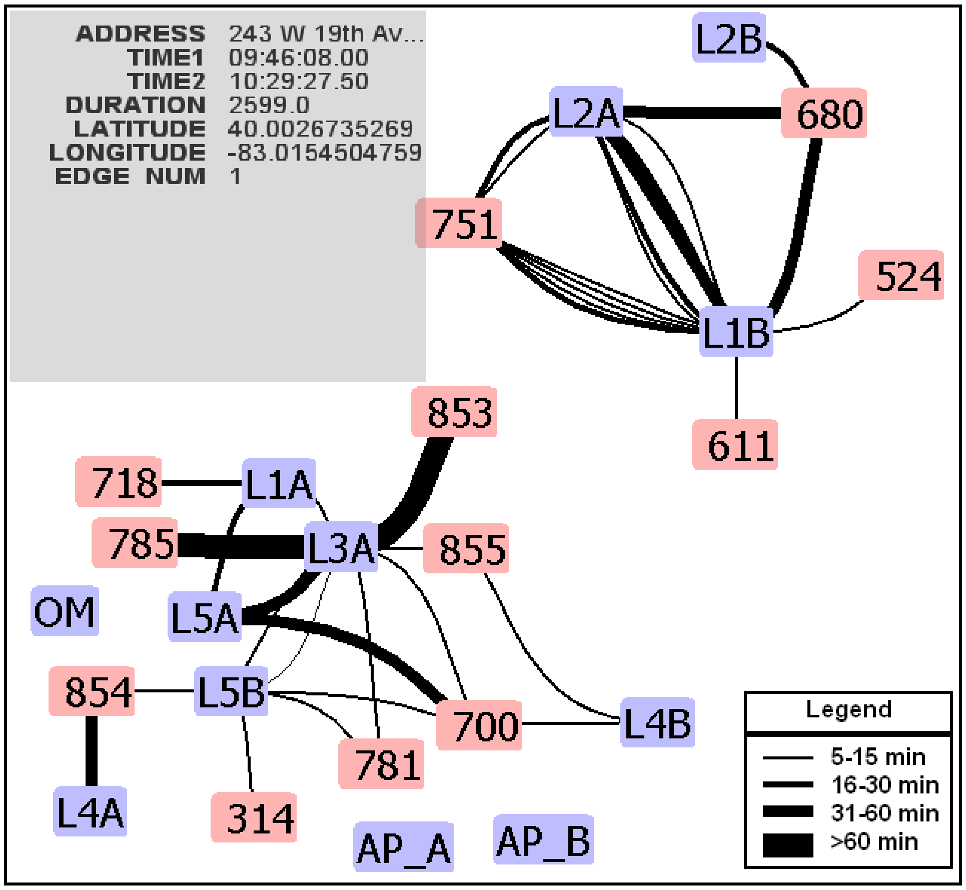
All the GPS data was loaded into Oracle Spatial by GPS track. The data was then processed with Oracle Spatial SQL code to determine where tracks stopped following the definition of a stop as defined in Section 5.3.2. This list of stops was listed chronologically. To populate the node table, we used SQL code to select the unique vehicle tracks from the results of the stop query. After the vehicles were added to the node table, we added those buildings that were the closest buildings to each stop. Static data like the latitude and longitude of buildings were also loaded to the node table at this time.

After the node table was fully populated, the edge table was populated. First vehicle-to-building edges were added by processing to stop query results. For each entry in the stop query results the criteria for vehicle-to-building links as defined in Section 5.3.2 was checked and if the time criteria was met, then a link was created between that track and the nearest building to that stop point. When the edge is created additional information like the start time, stop time, latitude, longitude, duration and address is loaded in the edge table. After the vehicle-to-building links were loaded, the vehicle-to-vehicle edges were loaded into the edge table. Again, the stop results were compared to itself returning those stops that occurred at the same location as defined in Section 5.3.3. The results represented those stops that occurred within 100 meters of each other. Those results were then processed to determine if there were overlapping times between two tracks and if so, were added to the edge table. Again, when the vehicle-to-vehicle edge is created additional information like the start time, stop time, latitude, longitude, duration and address is loaded in the edge table.

Once all the data was loaded in the node and edge tables, some additional code was written to improve the visualization. These included the ability to display information when selected, handle multiple links between nodes, and distinguish between short interactions and long interactions.

6.2 Results

The results of constructing the social network for the 14 GPS tracks are shown in Figure 6.1. Referring to Figure 6.1, each named node represents a vehicle track (blue), each numbered node represents a building (pink), and each link represents an interaction between vehicles and buildings that was at least five minutes long. Those links that are more heavily weighted represent durations that were greater than 15 minutes, greater than 30 minutes, and greater than 60 minutes. When a link is selected, the information about that interaction is displayed in the upper left corner. Multiple links represent multiple interactions. When a node is clicked on, an additional window is opened that looks like Figure 6.2. This window is another interactive window that allows a user to search for an actor or building by name. It also will animate and realign the graph around any node that is clicked on. This view is beneficial if there is interest on a certain individual already to see what connections exist for that particular individual.



ADDRESS	243 W 19th Av...
TIME1	09:46:08.00
TIME2	10:29:27.50
DURATION	2599.0
LATITUDE	40.0026735269
LONGITUDE	-83.0154504759
EDGE NUM	1

Figure 6.1: Social network graph of vehicle tracks and buildings

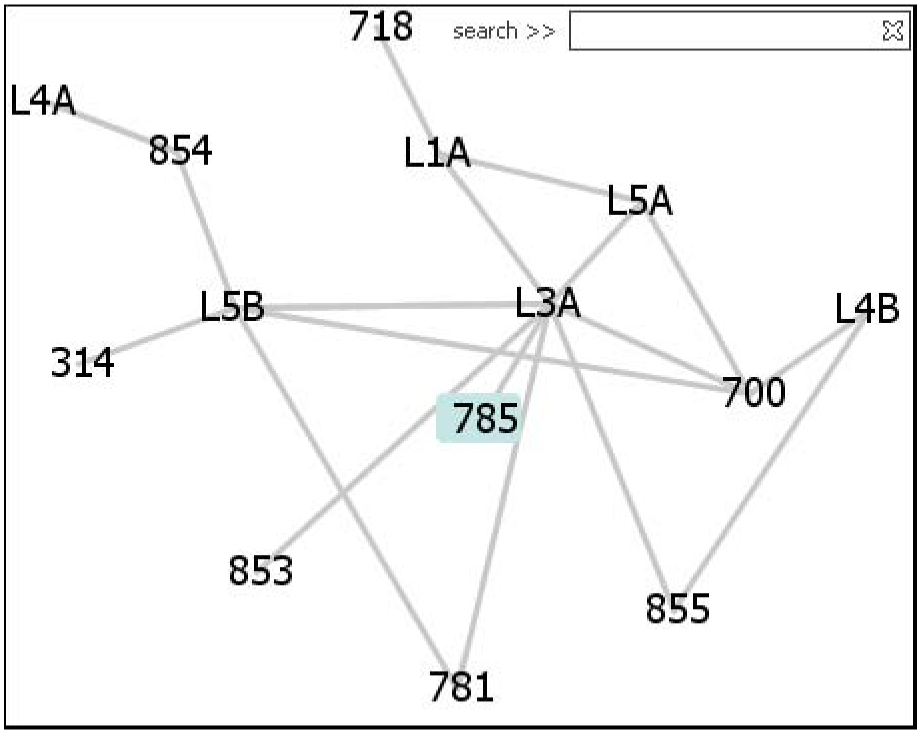


Figure 6.2: Radial view of social network graph

6.3 Enhancements

Some additional enhancements have been applied to this visualization. First, shown in Figure 6.3 is a social network, only with the buildings organized as they would be in geo-space. This view organizes the buildings in the same relative locations to one another as scaled by the size of the display window. This view provides us additional information of relativity and context. One item it shows is when buildings are very close to one another, this is depicted now visually where before it was unknown how far buildings are from one another. An example of this is in Figure 6.1 buildings 785, 700, 718, and 781 are clearly visible, while in Figure 6.3 they are so close to one another to the point that the label of building 785 obscures the labels for buildings 718 and 781. Another item it shows is the relative areas where actors are acting (i.e. acting in same area, or all over the city).

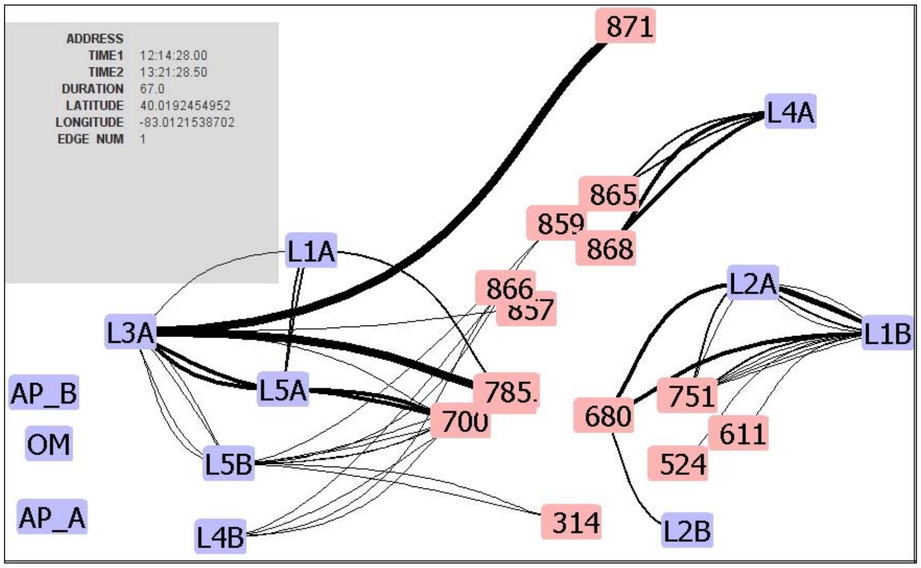


Figure 6.3: Social network with buildings organized in geo-space

6.4 Analysis

From these visualizations we can deduce information that was not known prior to the construction of the social network . The location represented by building 785 and 700 (middle of Figure 6.3) was an area of interest for drivers *L3A*, *L1A*, *L5A*, and *L5B* who all interacted with each other, and also all visited this same location. Relating to the map, the area that this refers to is a parking lot on the northeast side of the OSU football stadium. This parking lot was used as a coordination meeting place for the four actors prior to their activities.

The other group of interest represented by vehicles *L2A*, *L1B*, and *L2B* (right side, Figure 6.3) coordinated a separate activity. Specifically they coordinated a staged kidnaping event involving coordination between *L2A* and *L1B*. What is also depicted in the social network are two primary buildings of interaction, buildings 751 and 680. Figure 6.4 shows the GIS view of this area. After the social network was constructed it was verified that building 751 was a building of interest used as a staging area to time up a kidnapping event that occurred in the area of building 680.

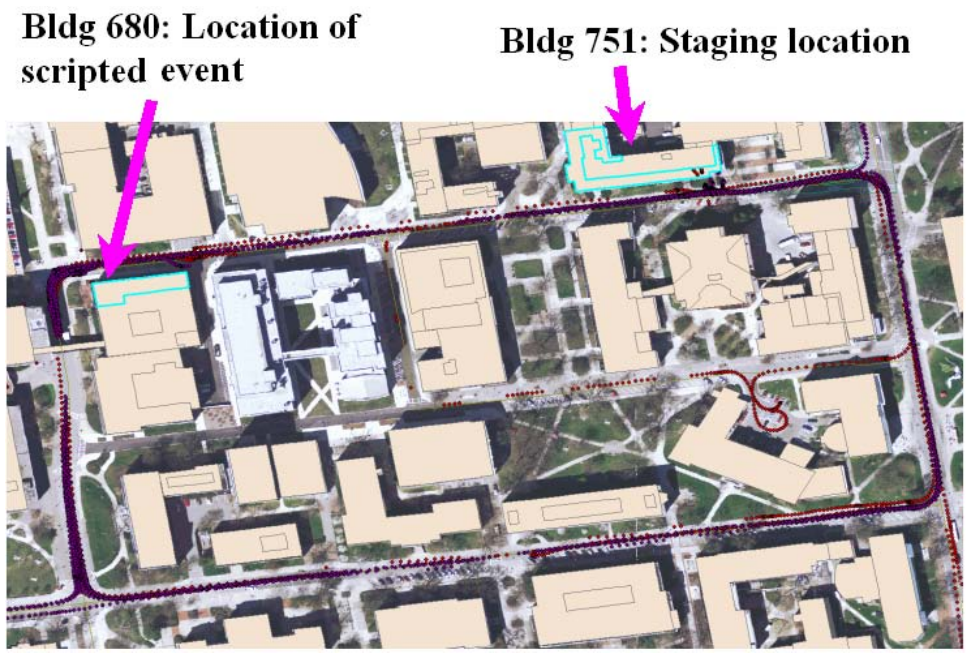
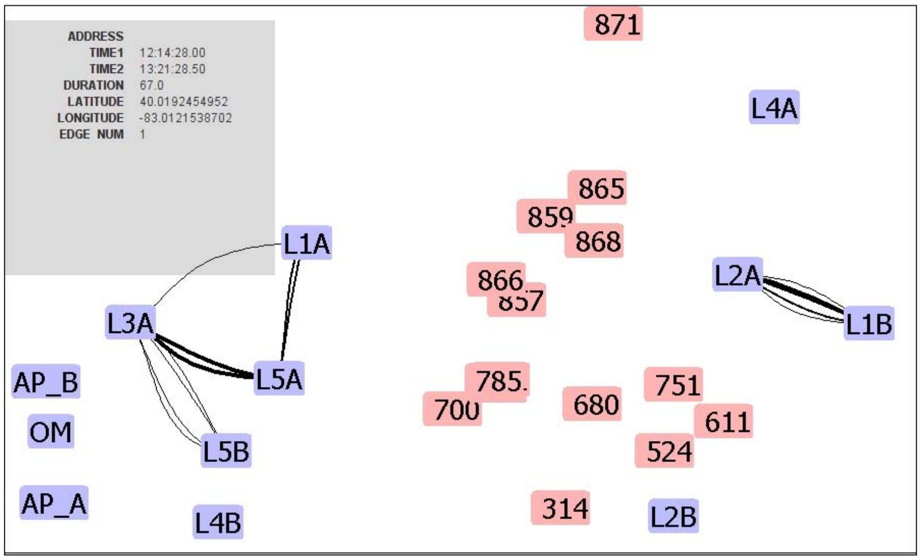

Bldg 680: Location of scripted event
Bldg 751: Staging location

Figure 6.4: Area of interest of kidnaping event

Also, referring back to the social graph in Figure 6.3 we are able to see that three tracks that were designated as independent random driving (AP_A, AP_B, and OM) did in fact have no interactions with anyone.

The detection of the interactions between actors when there was indeed interaction, and no links existing with actors that did not interact show that the social network is producing the desired results. It is able to relate vehicle-to-vehicle interactions and vehicle-to-building interactions effectively together. This provides us additional context to an unfolding situation.

Since other data sets are likely to be much larger than the data used, additional enhancements were made to the social network viewer. This included the ability to interactively filter out buildings and building interactions in order to focus primarily on vehicle-to-vehicle interactions. Figure 6.5 shows the social graph filtering out the vehicle-to-building links.



ADDRESS
TIME1 12:14:28.00
TIME2 13:21:28.50
DURATION 67.0
LATITUDE 40.0192454952
LONGITUDE -83.0121538702
EDGE NUM 1

Figure 6.5: Social graph with buildings links hidden

6.5 Summary

In this chapter we presented results on building a social network from vehicle tracks in an urban environment. We showed that the resulting output was useful in effectively relating actors together, and relating them to buildings of interest. By processing the information and displaying it in this manner we were able to determine which actors were acting in concert with one another and where they were working. Specifically we were able to determine which actors were random actors, which actors were working together on events. We also showed the staging location and location where a kidnapping event occurred. All this information was organized in a user friendly environment adaptable to the needs of an intelligence analyst.

VII. Conclusions

7.1 Research Goal

The goal of this research was to generate knowledge from persistent video without the constant intervention of a human. To meet this goal we first produced a system that detected a "casing event." With the *casing event* identifying an actor of interest (See Section 4.1.8) we built a social network of interactions that the actor of interest had with other vehicles and buildings in the video (See Section 6.4). In this manner we were able to move from syntactic lower-level data points, to understanding the behavior of vehicles, detecting casing events, and relating vehicles to one another. This processing was done without human intervention providing an analyst with the ability to focus on other critical tasks.

7.2 Results

Table 7.1: Results of Casing Detection

Line No.	**Inter-section Dist**	**Loop Dist**	**Inter-section On**	**Time match** (min)	$\mathbf{P_D}$(%)	**Prob Real** (%)
1	15m-10m	30m	N	3.75	100	91.5
2	15m-13m	30m	N	3.75	90	81.8

7.2.1 Casing Event Detection. In the area of event detection, we were able to detect all casing events in a data set that included eight different tracks of 17 hours of driving time and 43 casing events in an urban environment. Table 7.1 shows the configuration of the experiment and the results. All 43 casing events were detected and only 4 false positives were found.

The methodology of using a reasoner to detect a series of turns that have more than three turns in the same direction ignoring whether those turns are in an intersection. If a series of three turns is found, then begin looking back 3.75 min from turn one and comparing those detections within 15m and 10m of an intersection with those points 3.75 min after turn three and within 15m and 10m of an intersection. If a point-to-point comparison exists that is less that 30m away from each other, a casing event has been detected. Using this methodology we detected 100% of casing events (43 of 43 possible), and can say with 95% confidence that the probability that a detection is real falls between 83.54% and 99.46%. In this 100% detection mode (configured as Line #1 in Figure 7.1), 96 vehicles could be tracked simultaneously in real time. Using an intersection distance between 15m and 13m maintained a 90% probability of detection and 81.8% probability that a detection was real and sped up execution time to the point that 424 vehicles could be tracked simultaneously in real time.

Once a casing event was detected, additional analysis was be done to provide more information to the user of the system. This included a description of the time involved, a likely target based on the where the majority of time was spent, the number of times that vehicle has passed the location, and the address of the location of interest. These outputs can be used to help filter out those false alarms of individuals driving loops around the block, that are not actually casing a location.

Although the output of this effort was tabular, we took the next step in analysis. If an individual is indeed suspicious, more information would need to be gathered about their locations of operations and potential associates. We developed a social network visualization to represent the additional information that could be displayed from the existing data that has already been collected. This social network could be used to further confirm or refute the casing detection, or it could also add information about who the suspicious individual is working with.

7.2.2 Social Network Construction. In the social network area, we are able to construct a social network from a series of GPS tracks. This is shown in Figure 7.1 with the buildings organized as they would be in geo-space, that is to mean that they are in relative location to one another scaled by the size of the display window. In Figure 7.1, named nodes (blue) represent vehicle tracks, with numbered nodes (pink) representing buildings.

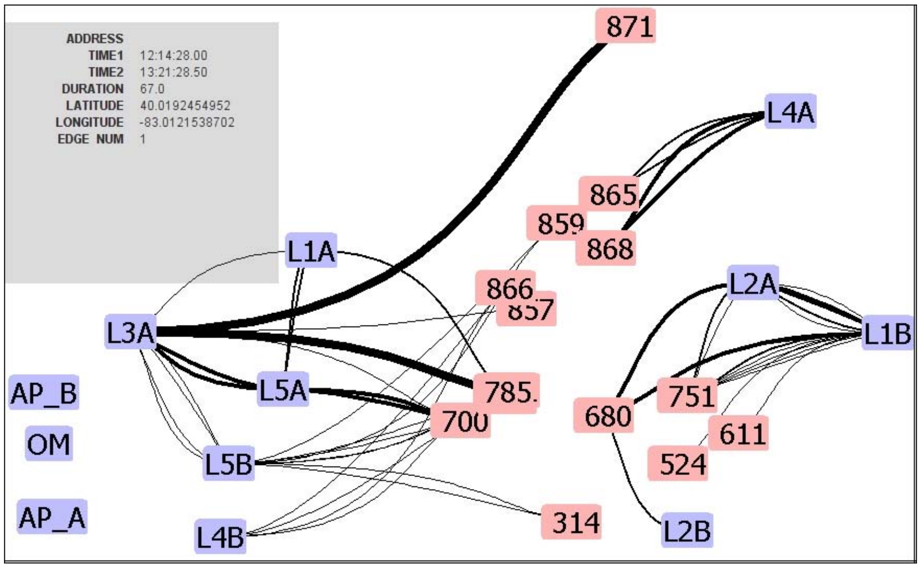

ADDRESS
TIME1 12:14:28.00
TIME2 13:21:28.50
DURATION 67.0
LATITUDE 40.0192454952
LONGITUDE -83.0121538702
EDGE NUM 1

Figure 7.1: Social network with buildings organized in geo-space

From this geographically organized depiction we were able to deduce information that was not known prior to the social network construction. The location represented by building 785 was an area of interest for drivers *L3A*, *L1A*, *L5A*, and *L5B* who all interacted with each other, and also all visited this same location. Relating to the map, the area that this refers to is a parking lot on the northeast side of the OSU football stadium. This parking lot was used as a coordination meeting place for the 4 actors prior to their activities.

The other group of interest represented by vehicles *L2A*, *L1B*, and *L2B* coordinated a separate activity. Specifically they coordinated a staged kidnaping event involving coordination between *L2A* and *L1B*. What is also apparent in the social network are two primary buildings of interaction, buildings 751 and 680. In Figure 7.2 it shows the GIS view of this area. After the social network was constructed it was verified that building 751 was a building of interest used as a staging area to time up a kidnapping event that occurred in the area of building 680. Also, referring back to the social graph in Figure 7.1 we are able to see that three tracks that were designated as independent random driving (*AP_A*, *AP_B*, and *OM*) did in fact have

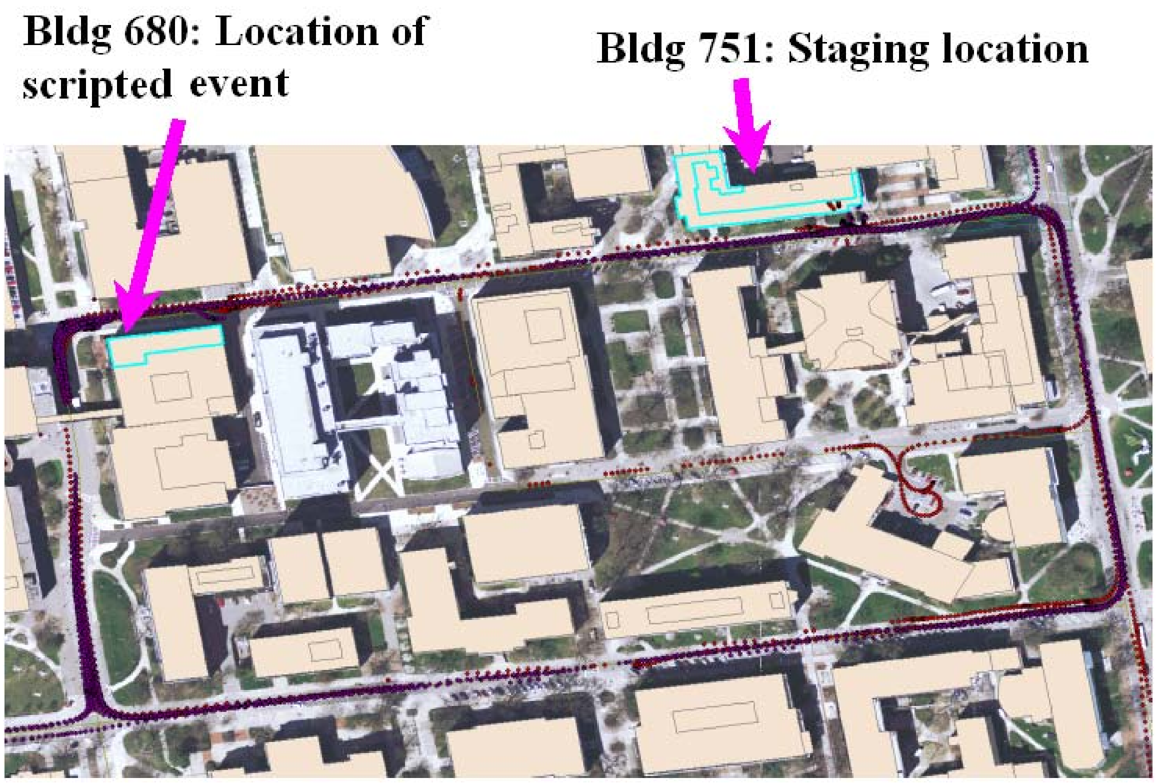

Bldg 680: Location of scripted event
Bldg 751: Staging location

Figure 7.2: Area of interest of kidnaping event

no interactions with anyone. The detection of these interactions show that the social network is able to pick out important events and relate them effectively together.

7.2.3 Results Summary. Putting the casing event detection and social network pieces together can be done with the kidnapping scenario that was enacted. While it was known ahead of time that some kind of activity was occurring, we were not aware who was involved ahead of time, when it would occur, or where it would occur. The casing event detector gives us the first clue as to who is involved, and where the event would occur. This is because a casing event triggered on vehicles *L1B* and *L2A*. The location that was predicted as the target was building 751, which ended up not being the target of the kidnapping, but instead was the location of the staging area prior to the capture. Once *L1B* and *L2A* are known as a vehicles of interest, we can determine that other actors are related to them because of their interaction in the social network. Referring to Figure 7.1, we can see that *L2A* had multiple interactions with *L1B*, and they both had interactions with building 751, the staging area. This evidence shows that no other actors were involved in the kidnapping event, except the possibility of *L2B*, who also had interaction with building 680 which both *L2A* and *L1B* had interaction with. At this point, the intelligence analyst knows that a suspicious event has likely occurred, who is involved, when it occurred, and where it occurred, all without reviewing any video of the area.

It is now the analysts choice whether he wants to go back and review the video to confirm the information, or collect more information about the suspicious event in order to identify what kind of event it was, or judge whether of not it was a true suspicious event. Regardless, the analyst can use the time stamp to review the video or the latitude/longitude to find the areas of interest in other tools.

7.3 Research Contributions

The contribution of this research is in closing the gap between raw data collection and human knowledge and understanding of persistent video. Where GIS systems can help somewhat visually and geographically orient someone to actions that occur, and allow us to correlate them in geo-space, more analysis is required to bring in time-dependent data to correlate it in both geo-space and time-space. This research focused on defining ways for geo-space and time-space to interact, and used example scenarios to show the benefit for the area of intelligence and surveillance.

The result is that a single analyst is freed up to track more than one event of interest at a time. The persistent video can collect on a wide area and feed it to the computer to detect the events of interest and build a network of interactions. When an event is detected, the analyst standing by can respond and verify in real time who the actor is, who they have been involved with, where they are currently, and where they have spent time in the past. The analyst now can choose whether the information is relevant and act on it if necessary.

7.4 Future Work

Surveillance is a critical function to all areas of military operations. Extracting intelligence from surveillance is a key link in orienting to the battlespace. The United States Air Force recognizes surveillance and reconnaissance as one of the seventeen key operational functions of air and space power. The suggestions are presented in the following three sections: *source data*, *event detection*, *extending network*.

7.4.1 Source Data. In order for the research accomplished in this thesis to be relevant, the tracking of vehicles from persistent video needs to be improved. A suggested accuracy needed is the ability to follow vehicles through turns for a continuous five minute period.

The casing event and social network algorithms can be used on a data set that is more rural. This can be used to determine how much the parameters need to vary from an urban to a rural area. In particular all the distance variables listed in Section 5.4 will likely need to be adjusted. If there are areas that do not have many buildings, or vary as to the density of the buildings widely, a area will have to be divided up into subareas to be useful in a social network. The integrating of a subdivided area and buildings alone would be a large sized undertaking.

7.4.2 Event Detection. The following items could be done in order to improve the accuracy of detecting true casing events, rather than just vehicles looping around buildings.

- A learning algorithm using the casing event detector to determine the best values for number of passes, time duration, percentage of building time, and amount of building time.
- Aggregating target building approach over several casing detections to improve accuracy in overall event detector. This could provide an overall cumulative baseline to compare individual events to in order to determine which locations routinely have traffic that is looping, and which do not.

- The addition of categorical information to the Jess reasoner to be able to predict what kind of surveillance is occurring, and also different method for predicting target of surveillance. This can include items such as an area or enemy that is known for striking certain targets categorically. Each building would be labeled categorically (i.e. transportation, restaurant, financial, parking lot) and the reasoner would recommend a target based on category, rather than amount of time spent.

The following items are bigger picture items that could be worked on in order to improve the overall concept of event detection through an automated process.

- Defining and testing normal and abnormal behavior. A system that could detect all abnormal behavior would be a great step forward. The problem is that the context of what normal behavior is for different situations has many different variables. Not only does the number of variables make the problem difficult, but the which variables are most important changes with the context.
- Adding more events to detect. These events might include brush passes, speeding on roads, running red lights, illegal parking, U-turns, driving the wrong way on one-way streets, driving on the wrong side of the road to mention a few.

7.4.3 Extending Network. The following items could be items of future work related to extending the social network.

- Adding the functionality to do Social Network Analysis or other analysis on the social network to determine who primary, secondary and central actors are.
- Augmenting the GIS-based social network with different types of connective information to perhaps include phone conversations, chat room interactions, social networking sites, or emails.
- Adding people to the social network. While this research focused on vehicles as primary actors, it is people that we are trying to model. Relating people

to vehicles, people to buildings, and people to other people would increase the usefulness of the resulting social network.

7.5 Summary

As a consequence of the research presented in this thesis, the detection of events using tracks generated from persistent video is possible but highly dependent on the ability of tracking objects from the video. Using GPS tracks the casing event detector detected 100% of casing events, and had a probability that a detection was real of 91.5%. A slight decrease in probability of detection to 90% decreases execution time by a factor of 4.4. In addition, tracks can be related to form social networks that define their patterns of behavior over time. Both the casing event detector, and the social network give us additional knowledge of the behavior of vehicles through automated processing techniques. If applied, these techniques can allow us to process more persistent video into intelligence faster, and with less human involvement.

Appendix A. Turn Detection

This appendix discusses the details of turn detection. It discusses the method that was used to attack the problem and presents and analyzes the results of experiments built from that methodology.

A.1 Methodology

Within the scope of detecting objects in surveillance video, there are many areas that can be improved. One main area of automated surveillance that can be improved upon is increasing the accuracy of event detection. In recent work individuals have been able to track moving vehicles with reasonable confidence [73] using a persistent video platform over Ohio State University (see Figure A.1) to track vehicle movements. To improve vehicle detection and event detection I will relate the detections together to determine if a vehicle is turning or not.

Figure A.1: A sample overhead surveillance shot over Ohio State University [63]

A.1.1 Goal. The goal is to detect events of interest in persistent video using a computer model and algorithm. The particular events of interest is the motion of vehicles, called "tracks." Table A.1 is a list of the events to be modeled and tested.

Using these tracked detections and other time and space information, a model will be developed to detect events of interest. The first step to accomplish this is to accurately determine if a track is turning.

Table A.1: The events that will be detected by the algorithm

Ref No.	**Name**	**Description**
1	Track Turn Left	A track was going in a certain direction, and now turned left
2	Track Turn Right	A track was going in a certain direction, and now turned right

A.1.2 Approach. The goal of this research is to augment persistent surveillance to detect events that would be of interest to a human surveillance monitor by finding events that are anomalies, or events that run contrary to expected behavior. The approach is to use overhead persistent surveillance video to detect those events of interest listed in Table A.1. Representative subsets of the OSU persistent video data set are used to test the event algorithm to determine if a detection occurs and compare those detections to whether the event actually occurred.

A.1.3 System Boundaries. The system that is being developed is called the Turn Detection System (TDS). A block diagram of the system is shown in Figure A.2. Along the left side with arrows into the system is the workload which will be discussed in Section A.1.5. The top arrows represent input parameters to the system and are discussed in detail in Section A.1.7. System components include: the physical computer where all the software is being run also known as the "subject computer," the Microsoft Access database software, the Visual Basic (VB) plug-in to Microsoft Access, and the component under test (CUT), the event detection algorithm.

A.1.4 System Services. TDS provides one service: Turn Detection. This service can have one of several outcomes, as listed in Table A.2. Referring to Table A.2, the first 4 outcomes are an exhaustive combination of what the system detects along

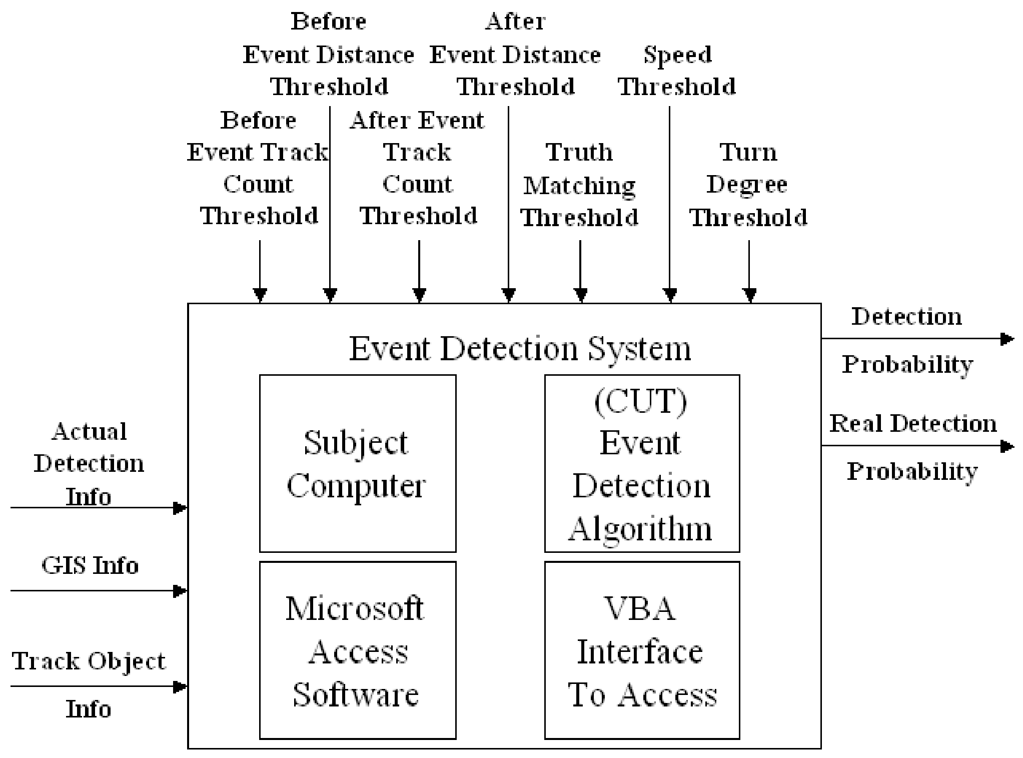

Before Event Distance Threshold
After Event Distance Threshold
Speed Threshold
Before Event Track Count Threshold
After Event Track Count Threshold
Truth Matching Threshold
Turn Degree Threshold
Event Detection System
Subject Computer
(CUT) Event Detection Algorithm
Microsoft Access Software
VBA Interface To Access
Actual Detection Info
GIS Info
Track Object Info
Detection Probability
Real Detection Probability

Figure A.2: A block diagram of the Turn Detection System

with what the truth of the situation is. If the system detects an event, and there is in fact that event occurring, the outcome will result in an overall detection or True Positive. Similarly, if the system detects an event, and there is not that event occurring, the outcome will result in a false alarm or False Positive. It follows that if the system does not detect an event that is in fact occurring, the outcome is a miss or False Negative. Lastly, if the system does not detect an event, and the event is not occurring, then the outcome is a non-detection or True Negative. A system that has high detections and non-detections, with low false alarms and misses is desirable.

It is assumed TDS will not malfunction. That is, the hardware and software will function correctly which is not the same as saying TDS will accurately detect turns. As a result, TDS determines a reading whether or not a turn has occurred, which direction the turn is, and is able to match that up with a truth value. Detections can

Table A.2: The possible outcomes of running the Turn Detection System

	System Detection	Truth	Description
detection	1	1	True Positive
false alarm	1	0	False Positive
miss	0	1	False Negative
non-detection	0	0	True Negative
false start			System does not start
no termination			System does not terminate
system fails			System terminates, but does not return yes or no

be used to form a probability of detection (True Positive), and a probability given the system detects an event, that it is a real event detection.

A.1.5 Workload. The workload is the Ohio State University (OSU) persistent video data shown in Figure A.1. Prior to input to the TDS, the video is processed to identify elements of the video to be denoted as tracks. In addition, the tracks are listed in a common coordinate system to relate to other static information from the Geographic Information System (GIS). The OSU data is an appropriate workload because it is persistent video data. It is also taken from an urban environment where several vehicles are driving around producing tracks.

There are some concerns with the results of processing the video into tracks. The method of detecting tracks from images is not without error. The data set [63] has 8140 detections (dots) of vehicle movements within a 69.8 second period. The best results achieved on the OSU data set detected 77.1% of vehicle movements with a 43.2% probability of a false alarm [73]. This imagery processed, with the error involved, was able to relate the 8140 vehicle detects into 730 tracks. It is uncertain what error is associated with the relationship of vehicle detections to tracks. Of those 730 tracks, only 72 of them are 20 detections in length or longer. None of the 730 tracks involve vehicles that are turning at street corners. This makes testing a turn detection system difficult using tracks derived directly from persistent video.

While research is ongoing and continues to improve on building tracks from persistent video data, there is a data set that represents what the intended results of the track generation from images. It is Global Positioning System (GPS) data.

GPS data [62] was collected over the same geographic region as the imagery. While not without its own errors (mostly driven by losing communication with a satellite in certain locations), GPS provides a baseline data set that can be used to test a turn detection system. It is believed that continued improvement of generating tracks from change detectioned images will improve towards the data fidelity that exists currently with GPS data.

The second element of the workload is the GIS data. The GIS data (buildings, roads) was built by a different person than the detection algorithm, so the event detection algorithm generation is independent of track detection, and the GIS data.

Figure A.3 shows an example of what the data looks like in the GIS. In the figure, the diamonds represent a vehicle that was driven while logging Global Positioning System (GPS) information every second or half second to create a track. The dots represent change detections that were discovered in processing images of the area. The GPS data (diamonds) represent 14 separate vehicles driving routes over a 25.5 hour period with over 140,000 detections.

The GPS data is a good representative of the persistent video data for several reasons. First, the time between detections is very similar. The GPS data logs position coordinates every half second or second. The results of processing persistent video data creates detections that are between 0.6 and 0.9 seconds apart. Second, neither data set is perfect in its detections. GPS data will sometimes lose signal if too many satellites are obstructed by buildings. This is similar to the persistent video data missing detections because of obstructions or other errors. As a result of these two similarities, the GPS data is deemed a good representative of the tracking that persistent video will be able to achieve as the processing of the data improves over time.

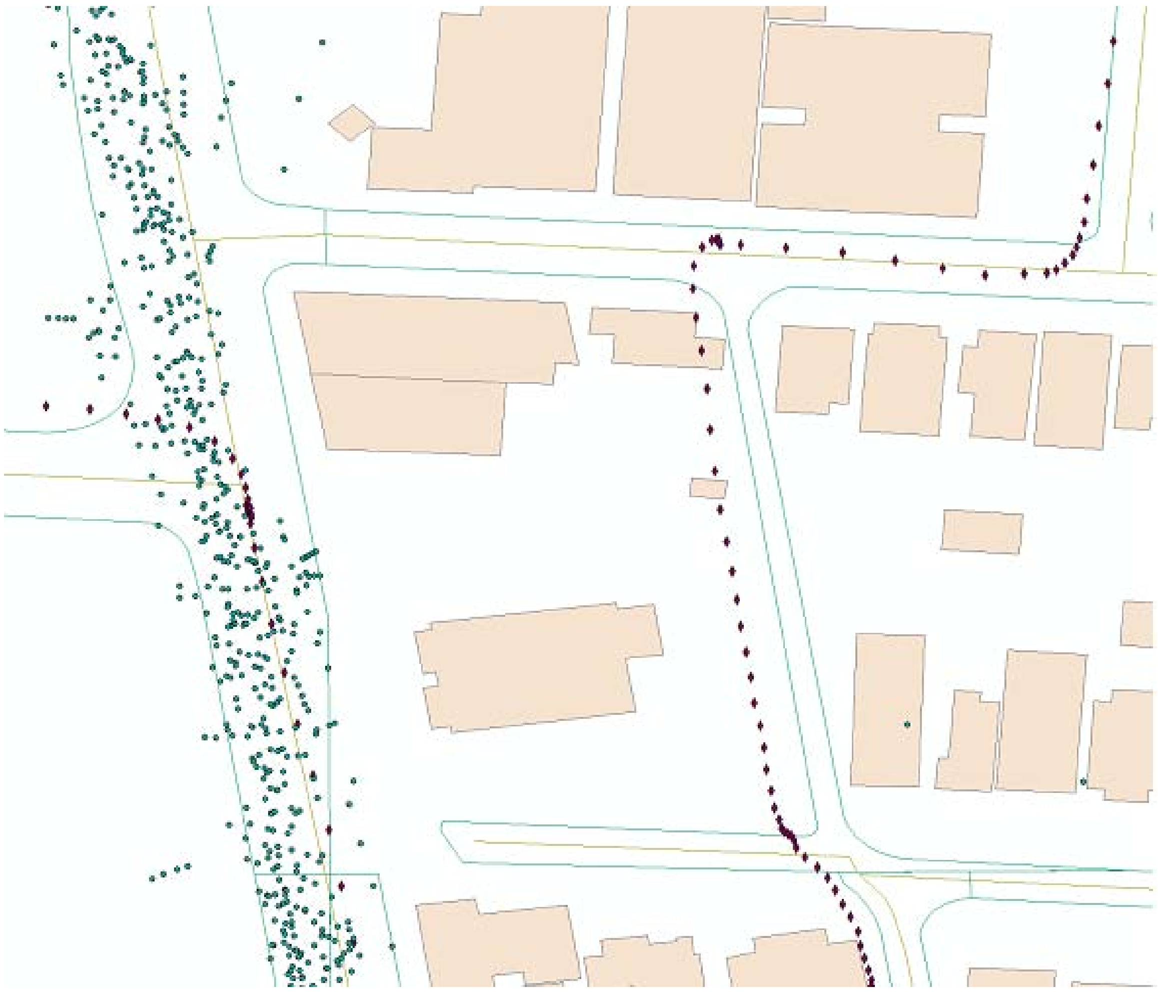


Figure A.3: Tracks viewed through the GIS

The final element of the workload is a data set of actual detections. It lists the X, Y coordinates along with T time of the event detection in the video data being run. This data is compared to the performance of the TDS to determine how accurate the TDS is in detecting turns.

Since turn detection is a subset of the overall events we intend to detect and model, we will only run tests on a portion of the 140,000 detections for turn detection analysis. Specifically we will run it against 3 of the 14 vehicles, representing 106 minutes of total driving time. This will leave data sets available for future testing against larger events.

A.1.6 Performance Metrics. The key element that is of interest is the accuracy of turn detection. A turn detection system that accurately detects turns with minimal false positives is useful, while one that misses turns is not useful neither is one that has many false positives. As a result, two metrics related to the accuracy of detection best determine whether the TDS is performing well. These are probability of detection and the probability that given a detection, a detection is real. Equations A.1 and A.2 are used to measure these metrics.

$$Probability\ of\ Detection(P_D) = \frac{Number\ of\ positives}{total\ number\ of\ actual\ true\ events} \tag{A.1}$$

$$P(RealTurn|DetectedTurn) = \frac{Number\ of\ positives}{Number\ of\ positives + Number\ of\ false\ positives} \tag{A.2}$$

A.1.7 System Parameters. Table A.3 lists the parameters of the TDS. For each track detection, the system detects whether an event of interest has occurred. The *Before Event Count Track Threshold* (called "*BE Count*" for brevity in discussion) establishes how far back the system looks to determine its history in terms of number of data points. These data points are the results of processing the images for motion detections at particular times and places. The *After Event Count Track Threshold* (*AE Count*) establishes how far in the future in terms of number of data points the system looks to determine if the event has occurred. Ultimately, the system checks to see if the event has occurred at the event point by establishing a trend from the past and comparing it to the future. The *BE Count* and *AE Count* thresholds define how far in the past or how far in the future. To help determine how many waypoints to look before and after, distance thresholds are established (*Before Event Distance Threshold* and *After Event Distance Threshold*). This makes the previous detection at least a certain distance from the point of interest instead of a certain number of data points. It counts back detections until at least the minimum distance is achieved above the *Before Event Distance Threshold*. The *After Event Distance* threshold is similar to the *Before Event Distance Threshold*, only it looks forward in time rather than backward.

The *Turn Degree Threshold* helps define what constitutes a turn. A turn that is less than the indicated threshold is defined as straight (slow, gradual turns less than the threshold will be missed). Lastly, the *Speed Differential Threshold* is used to determine the speed that the target is going into the turn minus the speed that the target is going after the turn. This differential of speed is used to test whether a target has slowed down prior to a turn, and accelerated after making the turn.

Figure A.4 shows an example of how these parameters are defined using the GPS data. In the figure, a vehicle is traveling south (down), turns right, and then goes west. There are two circles centered around the point of interest which represent the *Before Event Distance Threshold* and the *After Event Distance Threshold*. They determine the *Before Event Track Count* and *After Event Track Count* to be 7 and 6 respectively by selecting the first point that is outside their respective *distance thresholds*. Once these endpoints are chosen, the two resulting vectors form an angle which is measured using the dot product, and compared to the *Turn Degree Threshold* to determine if a turn has occurred or not. The *Speed Differential* is also calculated by taking the difference between the before turn speed and after turn speed. Both the before and after turn speeds are calculated from the distance vectors and divided by the change in time.

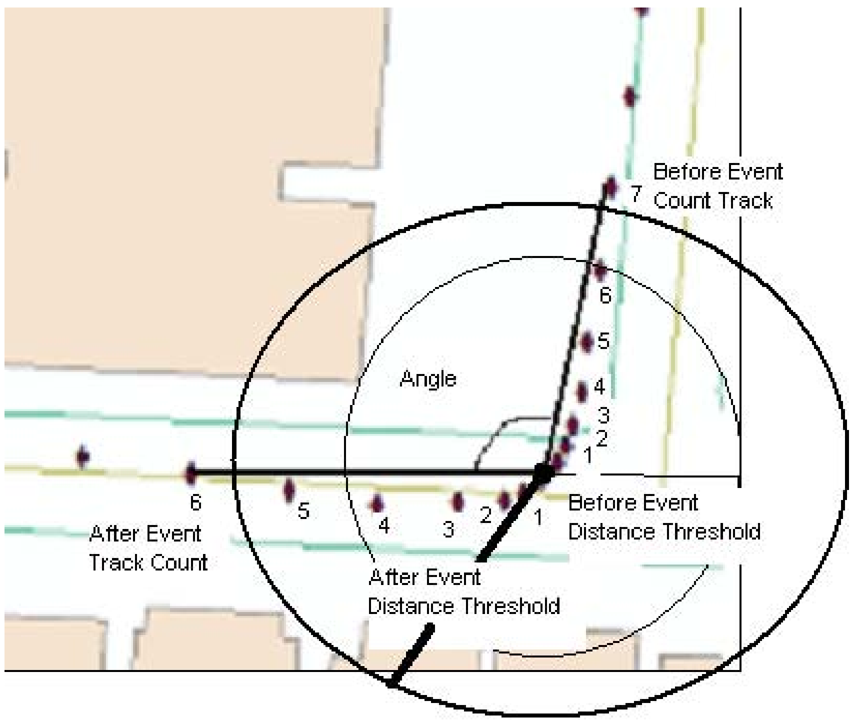


Figure A.4: Example of system parameters in the Turn Detection System

Table A.3: Parameters that affect the performance of the TDS

Name	**Description**
Before Event Track Count Threshold	The number of previous data points to be used to define the prior direction the target was headed.
After Event Track Count Threshold	The number of future data points to be used to define the new direction the target is headed.
Turn Degree Threshold	The threshold that will determine of what constitutes a turn in degrees.
Before Event Distance Threshold	The distance between the current data point and a former data point that is used to help determine which prior data point will be used to define the prior direction the target was headed.
After Event Distance Threshold	The distance between the current data point and a future data point that is used to help determine which future data point will be used to define the new direction the target is headed.
Matching Threshold	Given there is a event detected, how far should the algorithm look to find a match of truth before declaring it as a false detection.
Speed Differential Threshold	The difference between the speed which the target is leaving a turn minus the speed at which a target is approaching a turn.

A.1.8 Factors. The factors used in the experiment are the *Before Event Distance Threshold*, the *After Event Distance Threshold*, the *Turn Degree Threshold*, the *Matching Threshold*, and the *Speed Differential Threshold*. Each factor will be varied between experiments as shown in Tables A.4 - A.8. Table A.4 shows the levels that will be used for the *Before Event Distance* factor. Table A.5 shows the levels that will be used for the *After Event Distance* factor.

Table A.4: Levels of the *Before Event Distance* factor to be tested

Level	**Value**
1	35 meters
2	40 meters
3	45 meters
4	50 meters
5	55 meters

Table A.5: Levels of the *After Event Distance* factor to be tested

Level	**Value**
1	20 meters
2	25 meters
3	30 meters
4	35 meters
5	40 meters

Considering the potential input data, both the before event and after event have discrete data points. The approach taken is one that defines past and future not in terms of number of detection data points, but in terms of distance. Preliminary tests showed that looking out a greater distance in the past(before event) established a better initial direction vector, while a shorter after event turn vector detected turns with greater accuracy.

Table A.6 shows the levels of turn degrees that will be tested. Since most turns on roads deal with 90 degree angles, half that amount (45 degrees) was used in preliminary tests. After this initial experimentation is was determined that angles

Table A.6: Levels of the *Turn Degree* factor to be tested

Level	**Value**
1	20 degrees
2	25 degrees
3	30 degrees
4	37.5 degrees
5	45 degrees

smaller than 45 degrees improved detection, while false detections caused by curving roads could be limited with other factors.

The false detections of having such a small angle was minimized by collapsing continuous turn detections into one detection. An example of this might be explained using a 90 degree turn. It is very possible for several 20 degree turns to be detected centered on the 90 degree turn. A collapsing of these turns would take a range of one to many turns that are detected one after the other, and reduce it to one turn detection in the place of the middle point of all the continuous turns. An approach such as this would maximize detection by using a small angle, but minimize false detections by collapsing redundant detections together.

This collapsing of continuous turns is helpful in matching a detected turn with truth of actual turn detections as it gives only one detected turn to match with one truth of turn. Table A.7 shows the levels for the *Matching Threshold* factor. Because of the performance of vehicles turning normally taking about 2 to 5 seconds to complete, the matching of a turn detection to that the truth values should be in the same order of magnitude. If a matching threshold is too large, it is possible to confuse the system and miss detections as multiple turns might occur within the matching threshold buffer. This would cause turns to go undetected. At the same time, if the matching threshold is set to low, then the human error of lining up truth values exactly where the system detects a turn would be cumbersome to measure accurately. It is this discussion that drove the levels to be set as they are in Table A.7.

Table A.7: *Matching Threshold* time between a detection and the truth of an event occurring

Level	**Value**(seconds)
1	5
2	10
3	15

Table A.8: Threshold of *Speed Differential* between leaving a turn and entering a turn

Level	**Value** (miles per hour)
1	.5 mph
2	1 mph
3	2 mph
4	3 mph
5	4 mph

A.1.9 Evaluation Technique. In the experiments, I will be evaluating the accuracy of the TDS. Evaluation is necessary to determine if the accuracy of the system is acceptable and to provide a measurable level which can be improved upon if necessary. The technique used is direct measurement of the system because data for measurement is available (GPS tracks) and direct measurement is highly favored because of the increased believability in the results of the experiment. Table A.9 lists the particular configurations components in the TDS.

A.1.10 Experimental Design. Initially, a full factorial will be run using the levels discussed in Section A.1.8 without the speed threshold resulting in 375

Table A.9: Configuration of the TDS under test

Computer Specifications	Intel Xeon 3.2 GHz, 3 GB RAM
Operating System	Windows XP 64-bit Service Pack 2
Microsoft Access 2003	Service Pack 3
VBA Version	6.5
OSU data set collection dates	28 Apr 2006, 28 Oct 2007

tests. Then, once results are analyzed, one of the before or after thresholds will be fixed, a full factorial will be run varying the speed threshold, one of the before/after thresholds, the angle, and the matching threshold.

Only one replication will be run on each data set because the algorithm is deterministic, which means that given the same workload, the exact same results will be obtained. To get a reasonable level of confidence, it is important to have a workload that has enough detections. The data that will be used is an aggregation of 14 data sets comprising over 140,000 detections. Only 7% of the data has been tagged with truth values representing 123 turns. As stated earlier in Section A.1.5, only this 7% of the data will be used to configure and test the turn algorithm, leaving the rest of the data available for the testing of larger events.

The goal is to determine the probability of detection and probability a detection is true in order to apply it to future data sets. To do this, an interval is constructed based on ninety-five percent confidence. Ninety-five percent confidence is chosen because high confidence is expected for the data, but extreme confidence is not necessary (i.e. lives of people are not in jeopardy if the mean is improperly estimated).

A.1.11 Turn Detection Methodology Summary. Persistent surveillance is difficult because of the sheer amount of data that is collected. The goal of automated detection and reasoning is to improve the useability of the data that is collected. To move up the continuum from data to information ("things" to "worlds"), the first step is to determine that right configuration of factors to detect turns. The design of this experiment tests the factors of the events, specifically how far to look in the past, the future, and what angle constitutes a turn, and the speed differential to see which levels result in the highest probability of detecting a turn with the minimum probability of having a false positive result.

A.2 Turn Detection Results and Analysis

A.2.1 Algorithm. The overall process in detecting turns is shown in Figure A.5. Each point in a track sequence is run through the selection and detection process to produce a series of turn detections. Then, those detections are collapsed down if there are redundant contiguous detections. Lastly, the detections are compared against truth, and the calculations made for the probability of detection.

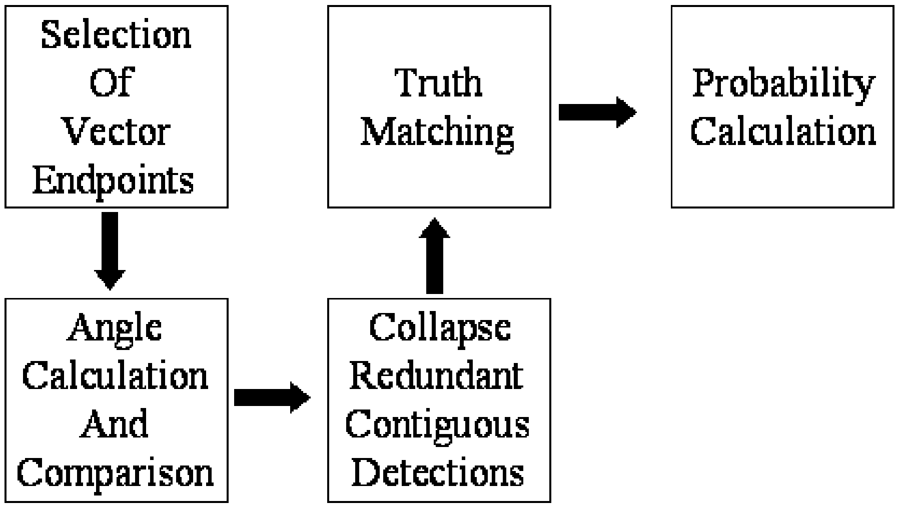

Selection Of Vector Endpoints
Truth Matching
Probability Calculation
Angle Calculation And Comparison
Collapse Redundant Contiguous Detections

Figure A.5: Block diagram of the sequence of steps in detecting turns

Algorithm A.2.1 details the first step in detecting turns, the selection of end points to create two vectors. It takes a track sequence and *Before* and *After distance thresholds* and determines the end points of the vectors. First, it determines an *Before Event Track Count* by looping and counting backwards until a point is reached that the distance between the point of interest and the *Before Event Track Point* is greater than the *Before Distance Threshold.* Then it determines the *After Event Track Count* by looping and counting forward until a point is reached that the distance between the point of interest and the *After Event Track Point* is greater than the *After Distance Threshold.* With the point of interest chosen, the *Before Event Track Point*, and the *After Event Track Point*, two vectors are created to be the vector inputs for Algorithm A.2.2. The first vector goes from the *Before Event Track Point* to the point of interest. The second vector goes from the point of interest to the *After Event Track Point*.

Algorithm A.2.1: SELECTION(
$TrackSequence, before\_distance\_threshold, after\_distance\_threshold$)

```
comment: to test all the data points, $i$ is iterated for each point in the track
for $i \leftarrow 1$ to LENGTH($TrackSequence$)
  do
  $Select P_i$
  $a \leftarrow i$
  $b \leftarrow i$
    while DISTANCE($P_i, P_b$) $< before\_distance\_threshold$
      do $b \leftarrow b - 1$
    while DISTANCE($P_i, P_a$) $< after\_distance\_threshold$
      do $a \leftarrow a + 1$
exit
```

Algorithm A.2.2 details the second step in detecting turns, the calculation of the angle between two vectors, the comparison verses the turn threshold, and a determination whether the turn is to the left or right. It does this by first computing the dot product between the two vectors resulting in the angle between the vectors. If the angle between the vectors is greater than the *Turn Degree Threshold* and the difference in speed of the before vector and the after vector is greater than the *speed differential* then a turn is detected. If a turn is detected, the algorithm uses the cross product between the two vectors to determine which direction the turn occurs. This direction of turn will become a basic semantic tag on the point of interest designating that the track is turning in a certain direction at that point.

Algorithm A.2.2: DETECTION($Vector1, Vector2, turn\_threshold, speed\_diff$)

```
if (DOT PRODUCT($Vector1$ to $Vector2$) $> turn\_threshold$ and
SPEED($Vector1$) $-$ SPEED($Vector2$) $> speed\_diff$)
  then
  {
    if (CROSS PRODUCT($Vector1$ to $Vector2$) $> 0$)
      then return (Left Turn)
      else return (Right Turn)
  }
  else return (No Turn)
exit
```

Algorithm A.2.3 details the third step in detecting turns, the collapsing of redundant turn detections that occur one after each other in the same direction into one turn detection centered on the middle of the series of redundant detections. This collapsing of redundant turns is helpful in comparing the TDS results to the truth value in order to determine how accurate the TDS is. Ordinarily as an output, the redundant turns will be kept in order to give more semantic information to a higher level of knowledge. In this way it will better represent true reality in which turns take time to occur rather than happening instantaneously. But for truth matching it is more accurate to reduce contiguous, redundant turns in the same direction down to one turn.

Algorithm A.2.3 is run on the full output of Algorithm A.2.2 taking the entire *Detection Sequence* as an input. It then processes through the sequence looking for detections to collapse. When it finds a detection, it counts the number of detections in the same direction and reduces them to one detection centered on the midpoint of the sequence of detections.

Algorithm A.2.3: COLLAPSEDETECTIONS(*DetectionSequence*)

```
for $i \leftarrow 1$ to LENGTH($DetectionSequence$)
  do
    $Overall\_Turn\_Direction \leftarrow Turn\_Direction_i$
    while $Overall\_Turn\_Direction = Turn\_Direction_i$
      do
        CLEAR($Turn\_Direction_i$)
        $i \leftarrow i + 1$
        $turn\_length \leftarrow turn\_length + 1$
    $Turn\_Direction_{i-turn\_length/2} \leftarrow Overall\_Turn\_Direction$
exit
```

Algorithm A.2.4 details the last step in detecting turns, the matching of truth value to collapsed detections. It takes the output of Algorithm A.2.3 and matches it to the truth values to determine whether a point is a detection, false alarm, miss, or non-detection. The algorithm assumes that if an item is not a detection, a false alarm, or a miss, then it is a non-detection (truth is 0, detection is 0 as in Table A.2). It then iterates through the joined tables of Truth and Detection matched up by point and time until it reaches a truth or detection. If it reaches a detection first, it then begins a countdown from the *matching threshold* to 0 to find a matching truth. If it finds one, then it denotes that detection as a true detection. If it does not find one, then it denotes the detection as a false alarm (False Detection). If it reaches a truth first, it begins a countdown from the *matching threshold* to 0 to find a matching detection. If it finds one, then it denotes the detection as a true detection. If it does not find one, then a missed detection has occurred (truth is 1, detection is 0 as in Table A.2).

Algorithm A.2.4: TruthMatch($DetectionSeq, matching\_threshold, TruthSeq$)

```
for $i \leftarrow 1$ to Length($DetectionSeq$)
  do
    if (Is_Turn($DetectionSeq_i$) $= TRUE$)
      then
        $Overall\_Turn\_Direction \leftarrow Turn\_Direction_i$
        $countdown \leftarrow matching\_threshold$
        while not ($countdown = 0$) and not $match\_found$
          do
            if ($Overall\_Turn\_Direction = Truth_i$)
              then
                $match\_found = TRUE$
                $True\_Detection_i = TRUE$
            $countdown \leftarrow countdown - 1$
            $i \leftarrow i + 1$
        if ( not $match\_found$)
          then $False\_Detection_{i-countdown} = TRUE$

      else
        if (Is_Turn($Truth_i$))
          then
            $Overall\_Turn\_Direction \leftarrow Truth_i$
            $countdown \leftarrow matching\_threshold$
            while not ($countdown = 0$) and not $match\_found$
              do
                if ($Overall\_Turn\_Direction = Turn\_Direction_i$)
                  then
                    $match\_found = TRUE$
                    $True\_Detection_i = TRUE$
                $countdown \leftarrow countdown - 1$
                $i \leftarrow i + 1$
            if ( not $match\_found$)
              then $Missed\_Truth_i = TRUE$
exit
```

Algorithms A.2.1, A.2.2, and A.2.3, when run one after the other take a track sequence along with the thresholds for *Before Event distance*, *After Event Distance*, *Turn Degree Angle*, and *Speed Differential*, produce a series of turn detections along the track sequence. The results of Algorithm A.2.4 are the counts of true and false detections, which can be used to calculate the probability of detection, and the probability that given we have a detection, the detection is real by following Equations A.1 and A.2.

Table A.10 shows an example results of Algorithm A.2.4. The results are that there were 107 detections, 12 false alarms, 16 missed detections, and 9,992 non-detections. Following Equation A.3, I step through the calculations to arrive at the probability of detection of 86.99% in Equation A.4. Following Equation A.5, I step through the calculations to arrive at the probability of given we have a detection, that it is a true detection of 89.92% in Equation A.4.

Table A.10: Results set for one combination of factors to demonstrate equation results

	Detection	**Non-Detection**
Turn	107	16
No Turn	12	9,992

$$Probability\ of\ true\ Positive = Sensitivity = \frac{Number\ of\ true\ positives}{total\ number\ of\ actual\ events} \tag{A.3}$$

$$Probability\ of\ true\ Positive = Sensitivity = \frac{107}{107+16} = 0.8699 \tag{A.4}$$

$$True\ detection\ rate = \frac{Number\ of\ true\ positives}{Number\ of\ true\ positives + Number\ of\ false\ detections} \tag{A.5}$$

$$True\ detection\ rate = \frac{107}{107+12} = 0.8992 \tag{A.6}$$

A.2.2 Results. Algorithms A.2.1, A.2.2, A.2.3, and A.2.4 were run against a 106 minute subset of 25.5 hours of GPS track data of vehicles driven on the OSU

Table A.11: Top results of Turn Detection without using a speed differential

Name	**Behind** (meters)	**Ahead** (meters)	**Angle** (degrees)	**Match** (sec)	**Prob Detect** (%)	**Prob Real** (%)
A-60-20-30-10	60	20	30	10	84.43	81.1
A-50-20-30-10	50	20	30	10	85.25	80
A-45-20-30-10	45	20	30	10	86.07	82.03
A-45-20-30-15	45	20	30	15	86.07	83.33

campus. Following is the sequence of experiments that were run, the results of each experiment, and the overall evolution toward finding the best configuration for turn detection.

A.2.2.1 Experiment #1: Finding the best before or after distance. Over the 106 minutes there were a total of 123 turns for the algorithms to detect. Table A.11 shows some of the highlights of the results of the first experiment, without using a speed threshold. As a result of these tests, it was determined that a good value for the ahead distance threshold was 20 meters, as that kept repeating in the top results, and also small angles like 30 degrees were producing the best results overall. These results led to an additional series of tests that fixed the angle threshold at 30 degrees, and the ahead distance at 20 meters in the search for an appropriate speed differential.

A.2.2.2 Experiment #2: Fixing ahead distance at 20 m and angle at 30 degrees. Since the Results of Experiment #1 led us to have top results with a 20 meter after event distance, Experiment #2 was designed to test the incorporation of speed differential into the results. The results of that experiment are shown in Table A.12.

The results in Table A.12 show a few general trends. First, as the speed differential is increased, the probability of detection decreases, and the probability given a detection, that it is a real detection increases. Since we wish to maximize both the

Table A.12: Initial results of Turn Detection using a speed differential

Name	**Bhd** (m)	**Ahd** (m)	**Ang** (deg)	**Match** (sec)	**Speed Diff** (mph)	**Prob Detect (%)**	**Prob Real (%)**
S-45-20-30-10-1	45	20	30	10	1	79.51	89.81
S-65-20-30-10-1	65	20	30	10	1	77.87	90.48
S-65-20-30-10-2	65	20	30	10	2	76.73	92.08
S-45-20-30-10-4	45	20	30	10	4	69.67	96.59
S-45-20-30-10-.5	45	20	30	10	0.5	81.15	89.19
S-45-20-30-10-.75	45	20	30	10	0.75	81.15	90.0
S-45-20-30-10-1	45	20	30	10	1	80.33	90.74

probability of detection and probability given we have a detection that the detection is real, a compromising value in between was arrived at with 0.5 mph speed differential. With the speed differential introducing a way to decrease false positives, at the cost of missing detections, it was thought that bringing in more detections through a smaller angle might improve overall performance. This led to the design and test of Experiment #3, to lower the turn angle even further to test this fact.

A.2.2.3 Experiment #3: Decreasing turn angle less than 30 degrees with speed differential. Experiment #3 was designed to see if the probability of detection might be increased by decreasing the turn angle and including the speed differential to minimize the additional false positives created by decreasing the turn angle. The results are shown in Table A.13. The results were positive and let to the acceptance of on overall best configuration of 45 meters behind (*Before Event Distance*, 20 meters ahead (*After Event Distance*, 20 degree turn angle, 10 second matching, and 0.5 mph speed differential.

A.2.3 Analysis. A binomial analysis was run on the results of configuration S-45-20-20-10-.5 (Last line in Table A.13) to determine a 95% confidence interval for detection of turns. The results of that analysis show that while we estimate the probability of detection at 86.99%, we can, with 95% confidence, say that it is between

Table A.13: Top results of Turn Detection using a speed differential

Name	**Bhd** (m)	**Ahd** (m)	**Ang** (deg)	**Match** (sec)	**Speed Diff** (mph)	**Prob Detect (%)**	**Prob Real (%)**
S-45-25-30-10-.5	45	25	30	10	0.5	82.79	89.38
S-45-20-30-10-.5	45	20	30	10	0.5	81.97	90.09
S-45-20-25-10-.5	45	20	25	10	0.5	86.07	89.74
S-45-20-20-10-.5	45	20	20	10	0.5	86.99	89.92

81.4% and 92.94%. This means that given another data set, we can expect 95% of the time that the average probability of detection will be between 81.4% and 92.94% on that new data set.

A binomial analysis was also run on the results of configuration S-45-20-20-10-.5 to determine a 95% confidence interval for probability that a detection is real. The results of that analysis show that while we estimate the probability a detection is real at 89.92%, we can, with 95% confidence, say that it is between 84.34% and 95.5%. This means that given another data set, we can expect 95% of the time that the average probability a detection is real will be between 84.34% and 95.5% on that new data set.

A.2.4 Turn Detection Summary. In this section I have demonstrated the ability to write and test an algorithm that detects turns for GPS vehicle tracks. By looking behind 45 meters, ahead 20 meters, with a 20 degree turn angle, 10 second matching, and 0.5 mph speed differential, I can say with 95% confidence that the system detects 81.4% to 92.94% of turns, and given the system detects a turn I can say with 95% confidence that it is a real detection between 84.34% and 95.5% of the time. While not without error, applying the TDS to the GPS data provides a step up the knowledge chain in semantics that are reasonably certain to be correct.

Appendix B. Oracle Spatial Semantic Tagging Code

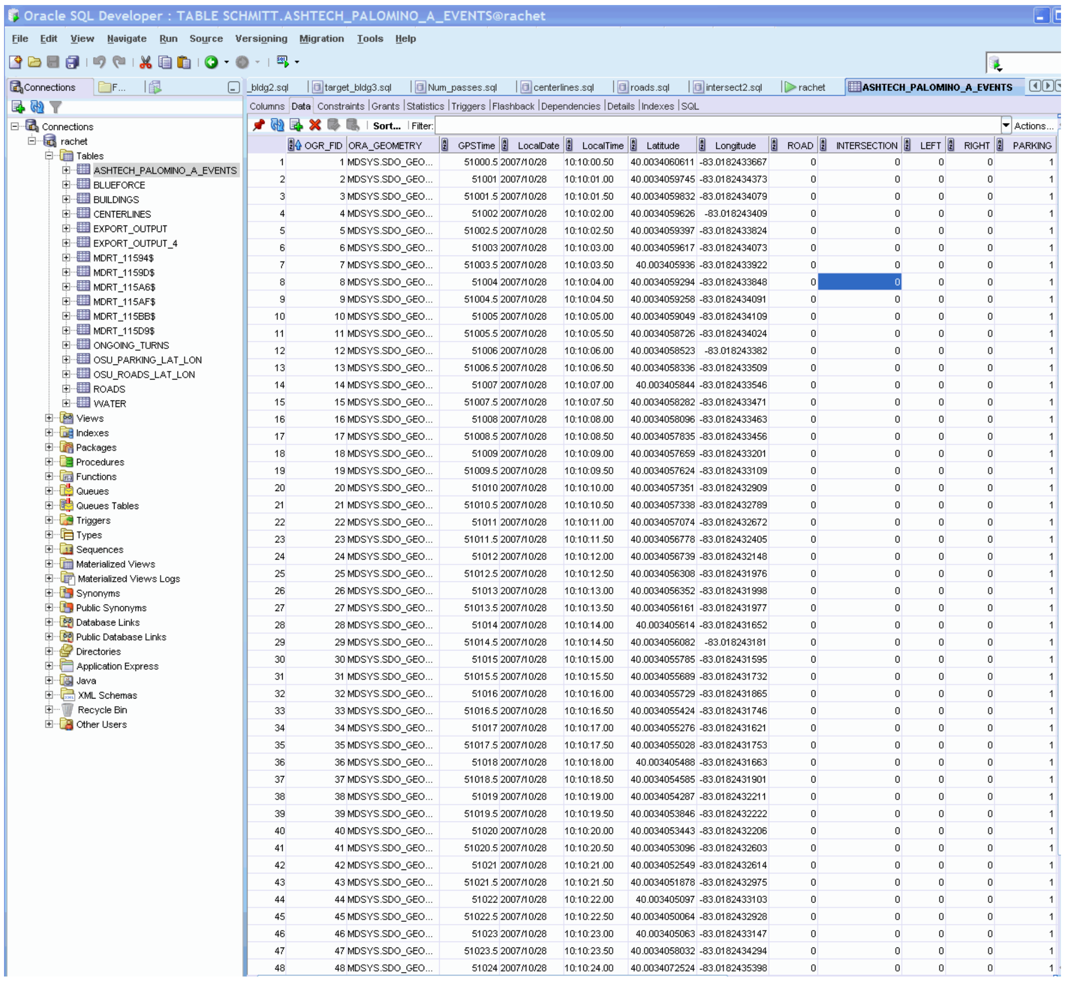
This appendix is intended to supplement the discussion of Oracle Spatial with specific code examples and results as it was applied to the GPS data and the campus of Ohio State University.

B.1 Database Organization

The data in the database is organized in a series of tables. The main data table has the GPS tracks and is called "ASHTECH_PALOMINO_A_EVENTS." A screen shot of this is shown in Figure B.1. In addition to the basic elements of time, latitude, and longitude, additional elements (road, intersection, left, right, parking) have been added to the table to be populated by the SQL code that will follow later. Each of these additional elements represent a truth value (1 for true and 0 for false) as to whether a waypoint is interacting with the element at that time and place. Also of note is the Geometry column, which is Oracle's interpretation of the record in geographical space. This geometry provides the means by which other layers (tables) can be compared to one another in geo-space. Lastly is the ID number of the data point which is used to distinguish between way points. Along the left side of Figure B.1 there is a listing of tables that will be reference later as well. An explanation of the pertinent tables and what data they hold is listed in Table B.1.

B.2 Road Semantics

The first element that will be updated will be associating roads with the GPS tracks. To do this, we will relate the *centerlines* table to the GPS track table with the sequence of statements that are listed in Table B.2. The first code sequence returns all GPS tracks that are within 15 meters of a centerline. The second code sequence removes the duplicate results of the first code sequence as a data point that is within 15 meters of two different centerlines would be returned twice. The third code sequence initializes the road value to false (0) on all GPS way points. The fourth code sequence sets the road value to true (1) whenever a way point is within 15 meters



	OGR_FID	ORA_GEOMETRY	GPSTime	LocalDate	LocalTime	Latitude	Longitude	ROAD	INTERSECTION	LEFT	RIGHT	PARKING
1	1	MDSYS.SDO_GEO...	51000.5	2007/10/28	10:10:00.50	40.0034060611	-83.0182433667	0	0	0	0	1
2	2	MDSYS.SDO_GEO...	51001	2007/10/28	10:10:01.00	40.0034059745	-83.0182434373	0	0	0	0	1
3	3	MDSYS.SDO_GEO...	51001.5	2007/10/28	10:10:01.50	40.0034059832	-83.0182434079	0	0	0	0	1
4	4	MDSYS.SDO_GEO...	51002	2007/10/28	10:10:02.00	40.0034059626	-83.018243409	0	0	0	0	1
5	5	MDSYS.SDO_GEO...	51002.5	2007/10/28	10:10:02.50	40.0034059397	-83.0182433824	0	0	0	0	1
6	6	MDSYS.SDO_GEO...	51003	2007/10/28	10:10:03.00	40.0034059617	-83.0182434073	0	0	0	0	1
7	7	MDSYS.SDO_GEO...	51003.5	2007/10/28	10:10:03.50	40.003405936	-83.0182433922	0	0	0	0	1
8	8	MDSYS.SDO_GEO...	51004	2007/10/28	10:10:04.00	40.0034059294	-83.0182433848	0	0	0	0	1
9	9	MDSYS.SDO_GEO...	51004.5	2007/10/28	10:10:04.50	40.0034059258	-83.0182434091	0	0	0	0	1
10	10	MDSYS.SDO_GEO...	51005	2007/10/28	10:10:05.00	40.0034059049	-83.0182434109	0	0	0	0	1
11	11	MDSYS.SDO_GEO...	51005.5	2007/10/28	10:10:05.50	40.0034058726	-83.0182434024	0	0	0	0	1
12	12	MDSYS.SDO_GEO...	51006	2007/10/28	10:10:06.00	40.0034058523	-83.018243382	0	0	0	0	1
13	13	MDSYS.SDO_GEO...	51006.5	2007/10/28	10:10:06.50	40.0034058336	-83.0182433509	0	0	0	0	1
14	14	MDSYS.SDO_GEO...	51007	2007/10/28	10:10:07.00	40.003405844	-83.0182433546	0	0	0	0	1
15	15	MDSYS.SDO_GEO...	51007.5	2007/10/28	10:10:07.50	40.0034058282	-83.0182433471	0	0	0	0	1
16	16	MDSYS.SDO_GEO...	51008	2007/10/28	10:10:08.00	40.0034058096	-83.0182433463	0	0	0	0	1
17	17	MDSYS.SDO_GEO...	51008.5	2007/10/28	10:10:08.50	40.0034057835	-83.0182433456	0	0	0	0	1
18	18	MDSYS.SDO_GEO...	51009	2007/10/28	10:10:09.00	40.0034057659	-83.0182433201	0	0	0	0	1
19	19	MDSYS.SDO_GEO...	51009.5	2007/10/28	10:10:09.50	40.0034057624	-83.0182433109	0	0	0	0	1
20	20	MDSYS.SDO_GEO...	51010	2007/10/28	10:10:10.00	40.0034057351	-83.0182432909	0	0	0	0	1
21	21	MDSYS.SDO_GEO...	51010.5	2007/10/28	10:10:10.50	40.0034057338	-83.0182432789	0	0	0	0	1
22	22	MDSYS.SDO_GEO...	51011	2007/10/28	10:10:11.00	40.0034057074	-83.0182432672	0	0	0	0	1
23	23	MDSYS.SDO_GEO...	51011.5	2007/10/28	10:10:11.50	40.0034056778	-83.0182432405	0	0	0	0	1
24	24	MDSYS.SDO_GEO...	51012	2007/10/28	10:10:12.00	40.0034056739	-83.0182432148	0	0	0	0	1
25	25	MDSYS.SDO_GEO...	51012.5	2007/10/28	10:10:12.50	40.0034056308	-83.0182431976	0	0	0	0	1
26	26	MDSYS.SDO_GEO...	51013	2007/10/28	10:10:13.00	40.0034056352	-83.0182431998	0	0	0	0	1
27	27	MDSYS.SDO_GEO...	51013.5	2007/10/28	10:10:13.50	40.0034056161	-83.0182431977	0	0	0	0	1
28	28	MDSYS.SDO_GEO...	51014	2007/10/28	10:10:14.00	40.003405614	-83.0182431652	0	0	0	0	1
29	29	MDSYS.SDO_GEO...	51014.5	2007/10/28	10:10:14.50	40.0034056082	-83.018243181	0	0	0	0	1
30	30	MDSYS.SDO_GEO...	51015	2007/10/28	10:10:15.00	40.0034055785	-83.0182431595	0	0	0	0	1
31	31	MDSYS.SDO_GEO...	51015.5	2007/10/28	10:10:15.50	40.0034055689	-83.0182431732	0	0	0	0	1
32	32	MDSYS.SDO_GEO...	51016	2007/10/28	10:10:16.00	40.0034055729	-83.0182431865	0	0	0	0	1
33	33	MDSYS.SDO_GEO...	51016.5	2007/10/28	10:10:16.50	40.0034055424	-83.0182431746	0	0	0	0	1
34	34	MDSYS.SDO_GEO...	51017	2007/10/28	10:10:17.00	40.0034055276	-83.0182431621	0	0	0	0	1
35	35	MDSYS.SDO_GEO...	51017.5	2007/10/28	10:10:17.50	40.0034055028	-83.0182431753	0	0	0	0	1
36	36	MDSYS.SDO_GEO...	51018	2007/10/28	10:10:18.00	40.003405488	-83.0182431663	0	0	0	0	1
37	37	MDSYS.SDO_GEO...	51018.5	2007/10/28	10:10:18.50	40.0034054585	-83.0182431901	0	0	0	0	1
38	38	MDSYS.SDO_GEO...	51019	2007/10/28	10:10:19.00	40.0034054287	-83.0182432211	0	0	0	0	1
39	39	MDSYS.SDO_GEO...	51019.5	2007/10/28	10:10:19.50	40.0034053846	-83.0182432222	0	0	0	0	1
40	40	MDSYS.SDO_GEO...	51020	2007/10/28	10:10:20.00	40.0034053443	-83.0182432206	0	0	0	0	1
41	41	MDSYS.SDO_GEO...	51020.5	2007/10/28	10:10:20.50	40.0034053096	-83.0182432603	0	0	0	0	1
42	42	MDSYS.SDO_GEO...	51021	2007/10/28	10:10:21.00	40.0034052549	-83.0182432614	0	0	0	0	1
43	43	MDSYS.SDO_GEO...	51021.5	2007/10/28	10:10:21.50	40.0034051878	-83.0182432975	0	0	0	0	1
44	44	MDSYS.SDO_GEO...	51022	2007/10/28	10:10:22.00	40.003405097	-83.0182433103	0	0	0	0	1
45	45	MDSYS.SDO_GEO...	51022.5	2007/10/28	10:10:22.50	40.0034050064	-83.0182432928	0	0	0	0	1
46	46	MDSYS.SDO_GEO...	51023	2007/10/28	10:10:23.00	40.003405063	-83.0182433147	0	0	0	0	1
47	47	MDSYS.SDO_GEO...	51023.5	2007/10/28	10:10:23.50	40.0034058032	-83.0182434294	0	0	0	0	1
48	48	MDSYS.SDO_GEO...	51024	2007/10/28	10:10:24.00	40.0034072524	-83.0182435398	0	0	0	0	1

Figure B.1: The table holding the basic GPS data

Table B.1: Listing of relevant tables in the Semantic tagging of GPS data

Table Name	**Description**
Ashtech_Palomino_A_events	The main table that holds all the GPS way points that when strung together in sequence create the track
Centerlines	Series of Centerlines of roads in the OSU area represented as lines
Buildings	Polygons that represent different buildings on the OSU campus
Ongoing Turns	Series of GPS data points with the results of Turn Detection
OSU Parking Lat Lon	Polygons that represent location and boundaries of parking lots in OSU

of a centerline. The end result when all the code is run in sequence is the GPS data table is updated with the semantics of whether a waypoint is on a road or not.

Table B.2: SQL code for updating road semantics

1	SELECT a.*
	 FROM ashtech_palomino_a_events a, centerlines r
	 WHERE SDO_WITHIN_DISTANCE
	 (a.ORA_GEOMETRY, sdo_cs.make_2d(r.ORA_GEOMETRY),
	 'DISTANCE=15 UNIT=METER ')='TRUE';

2	SELECT DISTINCT a.ogr_fid from ash_a_center_15 a,
	 ashtech_palomino_a_events B
	 WHERE A.ogr_FID = b.ogr_fid

3	UPDATE ASHTECH_PALOMINO_A_EVENTS SET road = 0

4	UPDATE ashtech_palomino_a_events
	 SET ashtech_palomino_a_events.road = 1
	 WHERE EXISTS
	 (SELECT *
	 FROM ash_a_center_15 A1
	 WHERE A1.ogr_fid = ashtech_palomino_a_events.ogr_fid)

B.3 Parking Lot Semantics

The second element that will be updated will be associating parking lots with the GPS way points. To do this, we will relate the parking lot table to the GPS track table with the sequence of statements that are listed in Table B.3. The first code sequence returns all GPS tracks that have any interaction with a parking lot polygon. This "any interaction" in the case of comparing points to polygons, includes any time the point is within the polygon, or on the edge (boundary) of the polygon. The second code sequence removes the duplicate results of the first code sequence. The third code sequence initializes the parking value to false (0) on all GPS waypoints. The fourth code sequence sets the parking value to true (1) whenever a waypoint interacts with any parking lot. The end result when all the code is run in sequence is the GPS data point table is updated with the semantics of whether each point is within a parking lot or not.

Table B.3: SQL code for updating parking lot semantics

1	SELECT a.ogr_FID
	FROM ashtech_palomino_a_events a, osu_parking_lat_lon R
	WHERE SDO_RELATE(a.ora_geometry, R.ora_geometry,
	'MASK=ANYINTERACT ') = 'TRUE'

2	SELECT DISTINCT a.ogr_fid
	FROM ash_a_PARKING a, ashtech_palomino_a_events B
	WHERE A.ogr_FID = b.ogr_fid

3	UPDATE ashtech_palomino_a_events SET parking = 0

4	UPDATE ashtech_palomino_a_events
	SET ashtech_palomino_a_events.PARKING = 1
	WHERE EXISTS
	(SELECT *
	FROM ash_a_PARKING A1
	WHERE A1.ogr_fid = ashtech_palomino_a_events.ogr_fid)

B.4 Turn Semantics

The next item that we will update semantics on is whether a way point is currently turning or not and in what direction. To illustrate this, I will just demonstrate the code for updating left turns as updating right turns is exactly the same. To do this we will compare the GPS data table with the *ongoing turns* table. The *ongoing turns* data was the results of the Turn Detection System with one change. In the turn detection, shown in Figure A.5 it was necessary for truth matching to collapse redundant contiguous turns to determine the accuracy of the turn detection system. When updating semantics, we are more interested in the series of way points that constitute the turn, not just the instantaneous point where the midpoint of the turn is. The series of way points is a better model of the actual behavior of a vehicle as it typically takes a vehicle a few seconds to complete a turn, rather than it being

Table B.4: SQL code for updating turn semantics

1	SELECT *
	FROM ongoing_turns a
	WHERE a."Turn" = 'L'

2	SELECT a.ogr_FID
	FROM ashtech_palomino_a_events a, ongoing_turn_L R
	WHERE SDO_RELATE(a.ora_geometry, R.ora_geometry,
	'MASK=ANYINTERACT ') = 'TRUE'

3	SELECT DISTINCT a.ogr_fid
	FROM turn_L a, ashtech_palomino_a_events B
	WHERE A.ogr_FID = b.ogr_fid

4	UPDATE ashtech_palomino_a_events SET left = 0

5	UPDATE ashtech_palomino_a_events
	SET ashtech_palomino_a_events.left = 1
	WHERE EXISTS
	(SELECT *
	FROM turn_L A1
	WHERE A1.ogr_fid = ashtech_palomino_a_events.ogr_fid)

an instantaneous change of direction. As a result, the ongoing turns represents the results of the turn detection system without collapsing turns.

The SQL code for updating turns is listed in Table B.4. The first code sequence returns all GPS tracks that are turning. The second code sequence matches the turning points with the points in the main GPS data table (*ashtech_palomino_a_events*). The third code sequence removes the duplicate results of the second code sequence. The fourth code sequence initializes the turn value (*left*) to false (0) on all GPS waypoints. The fifth code sequence sets the turn value (*left*) to true (1) whenever a waypoint has a turn that matches with that waypoint. The end result when all the code is run in sequence is the GPS data point table is updated with the semantics of whether each point is turning or not.

B.5 Intersection Semantics

The last item that we will update semantics on is whether a waypoint is in a road intersection or not. The SQL code for accomplishing this is listed in Table B.5. The first code sequence defines an intersection as the interaction between any two centerlines. In this way, if centerlines touch or cross, it will be labeled as an intersection. Once those combinations of street crossings are determined, the second code sequence checks each way point to see if it is within 15 meters of two different road centerlines. The third code sequence removes duplicate results. The fourth code sequence initializes the GPS waypoint values of intersection to false (0). Lastly, the fifth code sequence sets the intersection value to true (1) whenever a waypoint is within 15 meters of two centerlines that cross one another. The result when all sequences of code are run together is that the GPS waypoint data is updated with whether each waypoint is in an intersection or not.

Table B.5: SQL code for updating road intersection semantics

1	SELECT b.strname, C.strname as str_name_1, b.ora_geometry, c.ora_geometry as ora_geometry_1 FROM centerlines c, centerlines b WHERE SDO_RELATE(c.ora_geometry, b.ora_geometry, 'MASK=TOUCH+OVERLAPBDYINTERSECT +OVERLAPBDYDISJOINT') = 'TRUE' AND (b.ogr_fid <> c.ogr_fid)

2	SELECT a.ogr_FID
	FROM ashtech_palomino_a_events a, intersect_center i
	WHERE SDO_WITHIN_DISTANCE(a.ora_geometry,
	sdo_cs.make_2d(i.ora_geometry), 'DISTANCE=15 UNIT=METER ')
	= 'TRUE'
	AND SDO_WITHIN_DISTANCE(a.ora_geometry,
	sdo_cs.make_2d(i.ora_geometry_1), 'DISTANCE=15 UNIT=METER ')
	= 'TRUE'

3	SELECT DISTINCT * FROM ash_a_intersect
4	UPDATE ashtech_palomino_a_events a SET a.intersection = 0

5	UPDATE ashtech_palomino_a_events
	SET ashtech_palomino_a_events.intersection = 1
	WHERE EXISTS
	(SELECT *
	FROM ash_a_intersect A1
	WHERE A1.ogr_fid = ashtech_palomino_a_events.ogr_fid)

Bibliography

1. “2 Using Prolog’s Inference Engine”, 6/6/2008. URL `http://www.amzi.com/ExpertSystemsInProlog/02usingprolog.htm`.
2. “$20 - 30 - Lessthan50$ - General Purpose - Door Mats - Area Rugs & Mats - Flooring at The Home Depot”, 5/9/2008. URL `http://www.homedepot.com/`.
3. “About - Powerset”, 5/14/2008. URL `http://www.powerset.com/about/`.
4. “Air Force expands training program for Predator operators”, 11/26/2008. URL `http://www.nationaldefensemagazine.org/archive/2006/december/pages/ airforceexpand2779.aspx?pf=1`.
5. “ArcView Key Features”, 10/14/2008. URL `http://www.esri.com/software/arcgis/ arcview/about/features.html`.
6. “Bookmarks”, 10/14/2008. URL `http://www-sul.stanford.edu/depts/gis/bookmark.htm`.
7. “Cartography”, 10/14/2008. URL `http://www8.garmin.com/cartography/`.
8. “City of Wichita - Larceny Suspicious Activity Report”, 10/15/2008. URL `http://www.wichitagov.org/CityOffices/Police/Investigations/Larceny/Susp_Activity.htm`.
9. “cityofdenton.com: Suspicious Activities”, 10/15/2008. URL `http://www.cityofdenton.com/pages/policecrimesusp.cfm`.
10. “Connecting the Dots – Social Network Analysis of 9-11 Terror Network”, 11/18/2008. URL `http://www.orgnet.com/prevent.html`.
11. “craigslist classifieds: jobs, housing, personals, for sale, services, community, events, forums”, 5/12/2008. URL `http://www.craigslist.org/about/sites.html`.
12. “cwm - a general purpose data processor for the semantic web”, 5/30/2008. URL `http://www.w3.org/2000/10/swap/doc/cwm.html`.
13. “DAML+OIL (December 2000)”, 5/6/2008. URL `http://www.daml.org/2000/12/ daml+oil-index.html`.
14. “Dion Hinchcliffe’s Web 2.0 Blog”, 11/17/2008. URL `http://web2.socialcomputingmagazine.com/`.
15. “Discussion Topic: lane width”, 10/16/2008. URL `http://knowledge.fhwa.dot.gov/cops/`.
16. “Dublin Core Metadata Initiative (DCMI)”, 4/28/2008. URL `http://www.dublincore.org/`.

17. “Eclipse.org home”, 7/28/2008. URL http://www.eclipse.org/.

18. “Endeca Information Access Platform”, 5/14/2008. URL http://endeca.com/technology/technical-details.html.

19. “ESRI Products Overview”, 10/14/2008. URL http://www.esri.com/products/index.html.

20. “Exhibit - SIMILE”, 5/14/2008. URL http://simile.mit.edu/wiki/Exhibit.

21. “Faceted Browser - SIMILE”, 5/14/2008. URL http://simile.mit.edu/wiki/Faceted_Browser.

22. “Factsheets : MQ-1 Predator Unmanned Aircraft System : MQ-1 Predator Unmanned Aircraft System”, 12/23/2008. URL http://www.af.mil/factsheets/factsheet.asp?fsID=122.

23. “Factsheets : RQ-4 Global Hawk Unmanned Aircraft System : RQ-4 Global Hawk Unmanned Aircraft System”, 12/23/2008. URL http://www.af.mil/factsheets/factsheet.asp?fsID=13225.

24. “Flamenco Home”, 5/14/2008. URL http://flamenco.berkeley.edu/.

25. “GeoNames”, 10/30/2008. URL http://www.geonames.org/.

26. “Gottfried Wilhelm von Leibniz”, 5/30/2008. URL http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Leibniz.html.

27. “hakia Lab”, 5/14/2008. URL http://labs.hakia.com/hakia-lab-qdex.html.

28. “hakia Search Engine Beta”, 5/14/2008. URL http://company.hakia.com/technology.html.

29. “High-flying hunters: ground commanders can’t get enough of the Predator”, 11/26/2008. URL http://findarticles.com/p/articles/mi_m0IBP/is_3_51/ai_n27317767/pg_1?tag=artBody;col1.

30. “Homeland Security Today - news and analysis - Suspicious Behavior Could Indicate Terror Plotting”, 10/14/2008. URL http://www.hstoday.us/content/view/3932/128/.

31. “How the NSA does ”social network analysis.” - By Alexander Dryer - Slate Magazine”, 11/18/2008. URL http://www.slate.com/id/2141801/.

32. “Jena Semantic Web Framework”, 5/30/2008. URL http://jena.sourceforge.net/.

33. “Jess, the Rule Engine for the Java Platform”, 7/28/2008. URL http://herzberg.ca.sandia.gov/.

34. “Longwell - SIMILE”, 5/14/2008. URL http://simile.mit.edu/wiki/Longwell.

35. “Managing Your Database Using Oracle SQL Developer”, 10/14/2008. URL http://www.oracle.com/technology/obe/sqldev/sqldev.htm#t6.

36. “Mashup Dashboard - ProgrammableWeb”, 5/9/2008. URL http://www.programmableweb.com/mashups.

37. “Media Darling Powerset vs. Non-Media Darling Hakia”, 5/14/2008. URL http://whydoeseverythingsuck.com/2008/05/media-darling-powerset-vs-non-media.html.

38. “metamodel.com - What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model?”, 5/2/2008. URL http://www.metamodel.com/article.php?story=20030115211223271.

39. “National Institute of Justice”, 4/28/2008. URL http://www.ncfs.org/.

40. “Oracle Spatial, Locator, and Location-Based Services”, 11/24/2008. URL http://www.oracle.com/technology/products/spatial/index.html.

41. “PC AI - Expert Systems”, 11/24/2008. URL http://www.pcai.com/web/ai_info/ expert_systems.html.

42. “Pellet: The Open Source OWL DL Reasoner”, 5/30/2008. URL http://pellet.owldl.com/.

43. “prefuse — interactive information visualization toolkit”, 12/8/2008. URL http://prefuse.org/.

44. “ProgrammableWeb - Mashups, APIs, and the Web as Platform”, 5/9/2008. URL http://www.programmableweb.com/.

45. “The Protg Ontology Editor and Knowledge Acquisition System”, 5/30/2008. URL http://protege.stanford.edu/.

46. “Racer Systems GmbH & Co. KG : Products”, 5/30/2008. URL http://www.racer-systems.com/products/racerpro/features.phtml.

47. “Reporting Suspicious Activities — Police Department”, 10/15/2008. URL http://www.sandiego.gov/police/prevention/reportingterrorism.shtml.

48. “Shop HomeDepot.com for Appliances, Patio Furniture, Hampton Bay Lighting & Fans, Power Tools & much more”, 5/9/2008. URL http://www.homedepot.com/.

49. “SIMILE : Exhibit 2.0”, 5/12/2008. URL http://simile.mit.edu/exhibit/.

50. “Sociological Skills Used in the Capture of Saddam Hussein”, 11/18/2008. URL http://www2.asanet.org/footnotes/julyaugust05/fn3.html.

51. “Surveillance Surge”, 11/26/2008. URL http://www.mgt-kmi.com/mit-archive/10-mit-2008-archives-volume-12-issue-8/55-surveillance-surge.html.

52. “taxonomic - Definition from the Merriam-Webster Online Dictionary”, 4/28/2008. URL http://www.merriam-webster.com/dictionary/taxonomic.

53. "Vulnerabilities in the Terrorist Attack Cycle — Stratfor", 10/14/2008. URL `http://www.stratfor.com/vulnerabilities_terrorist_attack_cycle`.

54. "Web Ontology Language OWL / W3C Semantic Web Activity", 5/6/2008. URL `http://www.w3.org/2004/OWL/`.

55. "Welcome to Solr", 5/14/2008. URL `http://lucene.apache.org/solr/`.

56. "What Is GIS?", 10/14/2008. URL `http://www.gis.com/whatisgis/index.html`.

57. "What is URN?", 6/5/2008. URL `http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci214164,00.html`.

58. "[WorldCat.org] Search for books, music, videos, articles and more in libraries near you", 5/14/2008. URL `http://www.worldcat.org/`.

59. "AKT Reference Ontology", 2002. URL `http://www.aktors.org/publications/ontology`.

60. "Semantic Web Challenge", 2003. URL `http://challenge.semanticweb.org`.

61. *Army Field Manual 3-24. Counterinsurgency*. Headquarters Department of the Army, Washington D.C., 2006.

62. Air Force Research Laboratory, Sensors Directorate. "OSU GPS Data". 28 Oct 2007.

63. Air Force Research Laboratory, Sensors Directorate. "OSU Imagery data", 28 Apr 2006.

64. Alesso, H. Peter and Craig F. Smith. *Developing Semantic Web Services*. A K Peters, Ltd., Canada, 2005.

65. Alpaydin, Ethem. *Introduction to Machine Learning*. MIT Press, Cambridge, MA, 2004.

66. Berners-Lee, Tim, James Hendler, and Ora Lassila. "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities", 2001.

67. Brachman, R. J. and J. G. Schmolze. "An Overview of the KL-ONE Knowledge Representation System". *Cognitive Science 9*, 2:171, 1985.

68. Cardoso, Jorge. "The Semantic Web Vision: Where are We?" *IEEE Inteligent Systems*, Sept/Oct 2007:22, 2007.

69. Daconta, Michael C., Lee J. Obrst, and Kevin T. Smith. *The Semantic Web*. Wiley Publishing, Inc., Indaianapolis, Indiana, 2003. ISBN 0-471-43257-1.

70. Demsar, J. and F. Solina. "Using machine learning for content-based image retrieving". *International Conference on Pattern Recognition*, volume 3, 138. 1996.

71. Denny, Michael. "XML.com: Ontology Tools Survey, Revisited", July 14, 2004 2004. URL `http://www.xml.com/lpt/a/1447`.

72. Forgy, Charles. “Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem”. *Artificial Intelligence*, 19:17, 1982.

73. Fulton, Tom. “Change Detection for Processing of Angel Fire Imagery”, 2008.

74. Hanjalic, A. “Adaptive extraction of highlights from a sport video based on excitement modeling”. *IEE Transactions on Multimedia*, 7:1114, 2005.

75. Horn, Alfred. “On sentences which are true of direct unions of algebras”. *Journal of Symbolic Logic*, 16:14.

76. Huth, Michael and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, New York, 2006.

77. Kalyanpur, Aditya, Jennifer Golbeck, Jay Banerjee, and James Hendler. *OWL: Capturing Semantic Information using a Standardized Web Ontology Language*. Technical report, University of Maryland.

78. Minsky, M. *Semantic Information Processing*. MIT Press, Cambridge, MA, 1968.

79. Mittal, Ankush. “An Overview of Multimedia Content-Based Retrieval Strategies”. *Informatica*, 2006.

80. Moschella, David. “Semantic Applications, or Revenge of the Librarians”, 2003.

81. Naphade, M. R., T. Kristjansson, B. Frey, and T. S. Huang. “Probabilistic multimedia objects (multijects): A novel approach to video indexing and retrieval in multimedia systems”. *Proc. of ICIP*, 536. 1998.

82. Powers, Shelley. *Practical RDF*. O’Reilly and Associates, Inc., USA, 2003. ISBN 9780596002633.

83. Rasheed, Z., Y. Sheikh, and M. Shah. “On the use of computable features for film classification”. *IEEE Trans. Circuits Syst. Video Techn.*, 15:52, 2005.

84. Shadbolt, Nigel R., monica m.c. schraefel, Nicholas Gibbins, Hugh Glaser, and Stephen Harris. “CS AKTIVE Space: Representing Computer Science in the Semantic Web”. *2004 World Wide Web Conference*. ACM Press, 2004.

85. Smith, J. R. and S. F. Chang. “VisualSEEK: a fully automated content-based image query system”. *ACM Multimedia*, 87, 1996.

86. Torralba, A. B. and A. Oliva. “Semantic organization of scenes using discriminant structural templates”. *International Conference on computer vision*, volume 2, 1253. 1999.

87. Velastin, M. and S. A. Velastin. “Intelligent distributed surveillance systems: a review”. *IEE Proceedings*, 152(2):192, 2005.

88. Yang, Z. and C. C. J. Kuo. “A semantic classification and composite indexing approach to robust image retrieval”. *International Conference on Image Processing*, volume 1, 134. 1999.

89. Zhang, H. J., Y. Gong, S. W. Smoliar, and S. Y. Tan. "Automatic parsing of news video". *Int. Conf. On Multimedia Computing and Systems*, 45. May 1994 1994.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 21–03–2009	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2007 — Mar 2009

4. TITLE AND SUBTITLE

Automated Knowledge Generation
with
Persistent Surveillance Video

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Daniel T. Schmitt, Maj, USAF

5d. PROJECT NUMBER

09-200

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/09-06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory (Steve Rogers)
2241 Avionics Circle
WPAFB, OH 45433-7334
(DSN: 785-5668 x3305, steven.rogers@wpafb.af.mil)

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RY

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Air Force has increasingly invested in persistent surveillance platforms gathering a large amount of surveillance video. Ordinarily, intelligence analysts watch the video to determine if suspicious activities are occurring which is a time and manpower intensive process. Instead, this thesis proposes using tracks generated from persistent video, and building a model to detect events (a suspicious surveillance activity known as a casing event). To test our model we used Global Positioning System (GPS) tracks generated from vehicles driving in an urban area. The results show that over 400 vehicles can be monitored simultaneously in real-time and casing events are detected with high probability. In addition, persistent surveillance video is used to construct a social network from vehicle tracks based on the interactions of those tracks. Social networks that are constructed give us further information about the suspicious actors flagged by the casing event detector by telling us who the suspicious actor has interacted with and what buildings they have visited. The end result is a process that automatically generates information from persistent surveillance video providing additional knowledge and understanding to intelligence analysts about terrorist activities.

15. SUBJECT TERMS

persistent surveillance video, knowledge generation, reasoning engine, social network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Lt Col Stuart Kurkowski
U	U	U	UU	155	**19b. TELEPHONE NUMBER** *(include area code)* (937)255-3636, x7228; stuart.kurkowski@afit.edu